Motionnet®
RemoteI/O & RemoteMotion

Motionnet

DLLリファレンス

NPM
Nippon Pulse Motor Co., Ltd.

「ご使用に際して」

●弊社以外から購入の場合

弊社以外から製品を購入された場合の保証につきましては、購入先にお問い合わせください。

●保証期間

保証期間は、製品をご指定の場所に納入後、一年間と致します。

●保証範囲

本書に従った正常な使用状態のもとで、保証期間内に故障が発生した場合は、無償で修理を致します。
但し、保証期間内であっても、故障の原因が次のいずれかに該当する場合は、無償修理の対象外とさせて頂きます。

①当社又は当社が指定した者以外による改造又は修理に起因する場合
②納品後の落下、運送上での損傷に起因する場合
③製品の部品の自然劣化、摩耗又は疲労等による場合
（モータ軸受、歯車、グリス、ケーブル類等）
④当社製品の本来の使い方以外の使い方をしたことに起因する場合
⑤火災、地震、落雷、風水害、塩害、電圧異常その他の天災又は不可抗力に起因する場合
⑥その他、故障の原因が、当社の責とみなされない事由に起因する場合

注１）日本国外に輸出された製品に関しては、保証の対象外とさせて頂きます。
注２）無償修理は、弊社持ち込み修理のみとし、出張修理等は致しません。
注３）修理が行われた製品の保証期間は、故障前の保証期間を超えることなく、修理前の保証期間と同一になります。
注４）ここでの保証は、製品単体の保証を意味するものであり、製品の故障等により誘発される損害は保証の対象外とさせて頂きます。
注５）弊社の判断により、修理に代えて、交換とさせて頂く場合がございます。

●本書は、製品に含まれる機能詳細を説明するものであり、お客様の特定目的に適合することを保証するものではありません。
また、本取扱説明書に記載されている応用例、回路図等は参考用です。ご採用に際しては、機器・装置の機能や安全性をご確認のうえ、ご使用ください。

●弊社製品は、原則として、次のいずれかの用途にはご使用なさらないようお願い致します。ご使用を要望される場合には、必ず、弊社営業部にご相談ください。

①原子力設備、電力・ガス等の供給システム、交通機関、車両設備、各種安全装置、医療機器等の高い信頼性・安全性を必要とされる設備
②人命や財産に直接、危険が及ぶ可能性がある設備
③カタログ、取扱説明書等に記載のない条件や環境での使用

●弊社製品の故障により人命や財産に重大な損害が及ぼす可能性のある用途にご使用する場合には、冗長設計等により、必ず高い信頼性・安全性を確保してご使用ください。

目　　次

1. はじめに 1
1－1. 概要 1
1－2. 動作環境 1
1－3. DLL バージョン 1
1－4. 導入 2
1－4－1. デバイスドライバのインストール 2
1－4－2. プロジェクトへのファイルの組み込み 2
1－4－2－1. PPCI_L112 の場合 2
1－4－2－2. NPMCMNET-I/O104 の場合 4
1－4－2－3. MNET-PUSB3601 の場合 5
2. 機能構成 7
2－1. 機能分類 7
3. 関数一覧 8
3－1. センターボード機種別関数 8
3－1－1. PPCI-L112 ボード関数 8
3－1－2. NPMCMNET-I/O104 ボード関数 8
3－1－3. MNET-PUSB3601 ボード関数 8
3－2. センターボード共通関数 8
3－3. ローカルボード共通関数 8
3－4. ローカルボード機種別関数 9
3－4－1. ローカル１軸モーションボード関数 9
3－4－2. ローカル４軸(6143)モーションボード関数 10
3－4－3. ローカル４軸(6045)モーションボード関数 11
4. 基本関数 13
4－1. 基本関数使用上の注意 13
4－1－1. センターデバイス関連 13
4－1－2. ローカル１軸モーションデバイス関連 13
4－1－3. ローカル４軸(6045)モーションデバイス関連 13
4－2. 基本関数一覧 14
4－2－1. センターデバイス基本関数 14
4－2－2. １軸モーションデバイス基本関数 14
4－2－3. ４軸(6045)モーションデバイス基本関数 14

5. センターボードおよび、ローカルボード共通機能 15
5－1. 関数詳細 15
5－1－1. PPCI-L112 ボード関数 16
5－1－1－1. PPCI_L112Open 16
5－1－1－2. PPCI_L112Close 17
5－1－1－3. PPCI_L112GetParaIn 18
5－1－1－4. PPCI_L112GetParaOut 19
5－1－1－5. PPCI_L112SetParaOut 20
5－1－1－6. PPCI_L112SetCommSpeed 21
5－1－2. NPMCMNET-I/O104 ボード関数 23
5－1－2－1. NPMCMNETOpen 23
5－1－2－2. NPMCMNETClose 24
5－1－2－3. NPMCMNETGetParaIn 25
5－1－2－4. NPMCMNETGetParaOut 26
5－1－2－5. NPMCMNETSetParaOut 27
5－1－2－6. NPMCMNETSetCommSpeed 28
5－1－3. MNET-PUSB3601 ボード関数 30
5－1－3－1. MNETPUsbOpen 30
5－1－3－2. MNETPUsbClose 31
5－1－3－3. MNETPUsbSetCommSpeed 32
5－1－4. センターボード共通関数 34
5－1－4－1. MNETSysComm 34
5－1－4－2. MNETIOCommStart 35
5－1－4－3. MNETIOCommStop 36
5－1－4－4. MNETCenterReset 37
5－1－4－5. MNETGetBRKF 38
5－1－4－6. MNETGetEIOE 39
5－1－4－7. MNETGetIOPC 40
5－1－4－8. MNETGetSBSY 41
5－1－4－9. MNETGetDevType 42
5－1－5. センターデバイス基本関数 43
5－1－5－1. MNETSetCenterCmd 43
5－1－5－2. MNETGetCenterSts 44
5－1－5－3. MNETGetCenterIntSts 45

5－1－5－4. MNETWriteCenterRegister....46
5－1－5－5. MNETReadCenterRegister....47
5－1－6. ローカルボード共通関数....48
5－1－6－1. MNETSetLocalData....48
5－1－6－2. MNETGetLocalData....50
5－1－6－3. MNETSetDLLPar....52
5－2. 機能操作説明....53
5－2－1. ローカルデバイスのデータ構成....53
5－2－2. ローカルデバイスのデータ取得....54
5－2－3. ローカルデバイスのデータ設定....54
5－2－4. I/O ポート変化割込....55
5－2－4－1. I/O ポート変化割込の設定....55
5－2－4－2. I/O ポート変化割込の確認とクリア....56
5－2－5. 通信エラー....57
5－2－5－1. I/O 通信エラーフラグの確認とクリア....57
5－2－5－2. センターデバイス エラーカウンタの取得とクリア....58
5－2－6. 通信速度の設定....59
5－2－7. 全体の流れ....60
5－2－7－1. I/O デバイスプログラム概略....61
5－3. データ定義....63
5－3－1. ステータス／ビットマップ....63
5－3－1－1. センターデバイスステータス....63
5－3－1－2. センターデバイス割り込みステータス....64
5－3－2. 識別子....65
5－3－2－1. DLL 内部パラメータ項目番号定義....65
5－3－2－2. センターデバイスコマンド....65
5－3－2－3. センターデバイスレジスタコマンド....65
5－3－3. 戻り値....66
6. ローカル１軸モーションボード機能....67
6－1. 関数詳細....67
6－1－1. ローカル１軸モーションボード関数....67
6－1－1－1. MNETM1SetSpeed....67
6－1－1－2. MNETM1Inc....71
6－1－1－3. MNETM1Abs....73

6－1－1－4. MNETM1Continue 75
6－1－1－5. MNETM1Org 77
6－1－1－6. MNETM1EscOrg 79
6－1－1－7. MNETM1ToEL 81
6－1－1－8. MNETM1EscEL 83
6－1－1－9. MNETM1Ez 85
6－1－1－10. MNETM1Stop 87
6－1－1－11. MNETM1GetSpeed 88
6－1－1－12. MNETM1GetCounter 89
6－1－1－13. MNETM1SetCounter 91
6－1－1－14. MNETM1GetLtc 92
6－1－1－15. MNETM1GetBitDataStruct 94
6－1－1－16. MNETM1GetBitData 96
6－1－1－17. MNETM1SetBitData 98
6－1－1－18. MNETM1FixSpeedRange 99
6－1－1－19. MNETM1UnFixSpeedRange 100
6－1－1－20. MNETM1OutCsta 101
6－1－1－21. MNETM1OutCstp 102
6－1－1－22. MNETM1OvrideSpeed 103
6－1－1－23. MNETM1OvridePos 104
6－1－1－24. MNETM1ResetSEND 106
6－1－1－25. MNETM1SetPar 107
6－1－1－26. MNETM1GetPar 108
6－1－1－27. MNETM1SearchORG 110
6－1－1－28. MNETM1GetREST 112
6－1－1－29. MNETM1GetRIST 114
6－1－1－30. MNETM1Restart 116
6－1－1－31. MNETM1SoftPCS 118
6－1－1－32. MNETM1SoftLTC 119
6－1－1－33. MNETM1Reset 120
6－1－1－34. MNETM1SetDLLPar 121
6－1－1－35. MNETM1ReadParFile 122
6－1－1－36. MNETM1WriteParFile 123
6－1－2. ローカル 1 軸モーションデバイス基本関数 124

6－1－2－1. MNETM1WriteRegister 124
6－1－2－2. MNETM1ReadRegister 125
6－1－2－3. MNETM1WriteCommand 127
6－2. 機能操作説明 128
6－2－1. データ取得 128
6－2－1－1. 状態取得 128
6－2－1－2. カウンタ取得 130
6－2－1－3. 停止確認 130
6－2－1－4. エラー割り込み要因取得 131
6－2－1－5. イベント割り込み要因取得 131
6－2－2. スタートモード 132
6－2－3. 速度データ構造体 133
6－2－4. 加減速設定 137
6－2－4－1. 加減速指定方法 137
6－2－4－2. 加減速設定の自動補正 137
6－2－5. S 字動作設定 138
6－2－6. 原点復帰モード 139
6－2－6－1. 原点復帰モード:0 140
6－2－6－2. 原点復帰モード:1 142
6－2－6－3. 原点復帰モード:2 143
6－2－6－4. 原点復帰モード:3 144
6－2－6－5. 原点復帰モード:4 145
6－2－6－6. 原点復帰モード:5 146
6－2－6－7. 原点復帰モード:6 147
6－2－6－8. 原点復帰モード:7 148
6－2－6－9. 原点復帰モード:8 149
6－2－6－10. 原点復帰モード:9 149
6－2－6－11. 原点復帰モード:10 150
6－2－6－12. 原点復帰モード:11 150
6－2－6－13. 原点復帰モード:12 150
6－2－7. 原点サーチ動作 151
6－2－8. リスタート 152
6－2－9. 動作前確認 152
6－2－10. 速度設定に関する詳細事項 153

6－2－10－1. 速度倍率 153
6－2－10－2. FH 補正 154
6－2－11. SD 入力 155
6－2－12. PCS 入力 155
6－2－13. 同時スタート 156
6－2－14. 同時停止 156
6－2－15. 基本固定パラメータ 157
6－2－16. コンパレータ 157
6－2－17. ソフトリミット 160
6－2－18. イベント割り込み設定 161
6－2－19. ラッチ 162
6－2－20. 速度オーバーライド 163
6－2－21. 目標位置オーバーライド 163
6－2－22. 操作の流れ 164
6－2－22－1. プログラム概略 165
6－3. データ定義 168
6－3－1. ビットデータ構造体 168
6－3－2. ステータス／ビットマップ 172
6－3－2－1. メインステータス 172
6－3－2－2. エラー割り込み要因 172
6－3－2－3. イベント割り込み要因 173
6－3－3. 識別子 174
6－3－3－1. スタートモード 174
6－3－3－2. 動作方向 174
6－3－3－3. 加減速指定方法 174
6－3－3－4. S 字使用/未使用 174
6－3－3－5. カウンタ指定 174
6－3－3－6. ラッチ指定 175
6－3－3－7. 停止モード 175
6－3－3－8. 速度倍率レンジ 175
6－3－3－9. ビット選択 176
6－3－3－10. パラメータ項目番号定義 177
6－3－3－11. DLL 内部パラメータ項目番号定義 181
6－3－4. 戻り値 182

7. ローカル 4 軸(6143)モーションボード機能 183
7−1. 関数詳細 183
7−1−1. ローカル 4 軸(6143)モーションボード関数 184
7−1−1−1. MNETM4SetSpeed 184
7−1−1−2. MNETM4PtpInc 188
7−1−1−3. MNETM4LineInc 191
7−1−1−4. MNETM4Ptp 195
7−1−1−5. MNETM4Line 198
7−1−1−6. MNETM4Continue 202
7−1−1−7. MNETM4Org 205
7−1−1−8. MNETM4SetSpeedNo 208
7−1−1−9. MNETM4PtpIncNo 211
7−1−1−10. MNETM4PtpNo 214
7−1−1−11. MNETM4ContinueNo 217
7−1−1−12. MNETM4OrgNo 220
7−1−1−13. MNETM4StopChk 223
7−1−1−14. MNETM4GetREST 225
7−1−1−15. MNETM4GetRIST 227
7−1−1−16. MNETM4CancelPreReg 229
7−1−1−17. MNETM4Stop 230
7−1−1−18. MNETM4SetBitData 231
7−1−1−19. MNETM4GetBitData 232
7−1−1−20. MNETM4GetBitDataStruct 234
7−1−1−21. MNETM4GetSpeed 236
7−1−1−22. MNETM4GetCounter 237
7−1−1−23. MNETM4SetCounter 239
7−1−1−24. MNETM4GetLtc 241
7−1−1−25. MNETM4SoftLTC 243
7−1−1−26. MNETM4SoftPCS 244
7−1−1−27. MNETM4OutCsta 245
7−1−1−28. MNETM4OutCstp 246
7−1−1−29. MNETM4SetPar 247
7−1−1−30. MNETM4GetPar 249
7−1−1−31. MNETM4SetDLLPar 251

7－1－1－32. MNETM4Reset.....253
7－1－1－33. MNETM4ReadParFile.....254
7－1－1－34. MNETM4WriteParFile.....255
7－2. 機能操作説明.....256
7－2－1. データ取得.....256
7－2－1－1. 状態取得.....256
7－2－1－2. カウンタ取得.....258
7－2－1－3. 停止確認.....258
7－2－1－4. エラー割り込み要因取得.....258
7－2－1－5. イベント割り込み要因取得.....258
7－2－2. スタートモード.....259
7－2－3. 速度データ構造体.....260
7－2－4. 移動データ構造体.....264
7－2－5. 加減速設定.....266
7－2－5－1. 加減速指定方法.....266
7－2－5－2. 加減速設定の自動補正.....266
7－2－6. S 字動作設定.....267
7－2－7. 原点復帰モード.....268
7－2－7－1. 原点復帰モード:0.....268
7－2－7－2. 原点復帰動作モード:1.....270
7－2－8. 動作前確認.....271
7－2－9. 速度設定に関する詳細事項.....272
7－2－9－1. 速度倍率.....272
7－2－9－2. FH 補正.....273
7－2－10. SD 入力.....274
7－2－11. PCS 入力.....274
7－2－12. 同期スタート.....275
7－2－13. 同時停止.....276
7－2－14. 基本固定パラメータ.....276
7－2－15. コンパレータ.....277
7－2－16. イベント割り込み設定.....278
7－2－17. ラッチ.....278
7－2－18. 直線補間.....279
7－2－19. 次動作設定.....279

7－2－20. 操作の流れ 280
7－2－20－1. プログラム概略.....281
7－3. データ定義..... 285
7－3－1. ビットデータ構造体 285
7－3－2. ステータス／ビットマップ 289
7－3－2－1. エラー割り込み要因.....289
7－3－2－2. イベント割り込み要因.....289
7－3－3. 識別子 290
7－3－3－1. スタートモード.....290
7－3－3－2. 動作方向 290
7－3－3－3. 加減速指定方法.....290
7－3－3－4. S 字使用/未使用 290
7－3－3－5. カウンタ指定 290
7－3－3－6. ラッチ指定 291
7－3－3－7. 停止モード 291
7－3－3－8. 速度倍率レンジ 291
7－3－3－9. ビット選択 292
7－3－3－10. パラメータ項目番号定義 293
7－3－3－11. DLL 内部パラメータ項目番号定義 295
7－3－4. 戻り値 296
8. ローカル 4 軸(6045)モーションボード機能..... 297
8－1. 関数詳細 297
8－1－1. ローカル 4 軸(6045)モーションボード関数.....298
8－1－1－1. MNET6045SetSpeed 298
8－1－1－2. MNET6045PtpInc.....302
8－1－1－3. MNET6045Ptp.....304
8－1－1－4. MNET6045LineInc.....306
8－1－1－5. MNET6045Line 309
8－1－1－6. MNET6045ArcInc.....312
8－1－1－7. MNET6045Arc.....315
8－1－1－8. MNET6045Continue 318
8－1－1－9. MNET6045Org 320
8－1－1－10. MNET6045EscOrg.....322
8－1－1－11. MNET6045SearchOrg 324

8－1－1－12. MNET6045ToEL....326
8－1－1－13. MNET6045EscEL....328
8－1－1－14. MNET6045Ez....330
8－1－1－15. MNET6045Stop....332
8－1－1－16. MNET6045SetSpeedNo....333
8－1－1－17. MNET6045PtpIncNo....337
8－1－1－18. MNET6045PtpNo....339
8－1－1－19. MNET6045ContinueNo....341
8－1－1－20. MNET6045OrgNo....343
8－1－1－21. MNET6045EscOrgNo....345
8－1－1－22. MNET6045SearchOrgNo....347
8－1－1－23. MNET6045ToELNo....349
8－1－1－24. MNET6045EscELNo....351
8－1－1－25. MNET6045EzNo....353
8－1－1－26. MNET6045StopNo....355
8－1－1－27. MNET6045GetBitStruct....356
8－1－1－28. MNET6045GetBit....358
8－1－1－29. MNET6045SetBit....360
8－1－1－30. MNET6045GetSpeed....361
8－1－1－31. MNET6045GetCounter....363
8－1－1－32. MNET6045SetCounter....365
8－1－1－33. MNET6045Reset....366
8－1－1－34. MNET6045CancelPreReg....367
8－1－1－35. MNET6045StopChk....368
8－1－1－36. MNET6045SetDLLPar....370
8－1－1－37. MNET6045SetPar....372
8－1－1－38. MNET6045GetPar....373
8－1－1－39. MNET6045FixSpeedRange....375
8－1－1－40. MNET6045UnFixSpeedRange....376
8－1－1－41. MNET6045SetFASpeed....377
8－1－1－42. MNET6045OvrideSpd....378
8－1－1－43. MNET6045GetRemain....380
8－1－1－44. MNET6045GetLtc....382
8－1－1－45. MNET6045SoftLtc....384

8－1－1－46. MNET6045SoftPcs 385
8－1－1－47. MNET6045OutCsta 386
8－1－1－48. MNET6045OutCstp 388
8－1－1－49. MNET6045GetTermAbs 389
8－1－1－50. MNET6045OvridePos 391
8－1－1－51. MNET6045ReadParFile 392
8－1－1－52. MNET6045WriteParFile 393
8－1－1－53. MNET6045PulserPtpInc 394
8－1－1－54. MNET6045PulserPtp 396
8－1－1－55. MNET6045PulserLineInc 398
8－1－1－56. MNET6045PulserLine 401
8－1－1－57. MNET6045PulserArcInc 404
8－1－1－58. MNET6045PulserArc 407
8－1－1－59. MNET6045PulserContinue 410
8－1－1－60. MNET6045PulserPtpIncNo 412
8－1－1－61. MNET6045PulserPtpNo 414
8－1－1－62. MNET6045PulserContinueNo 416
8－1－1－63. MNET6045GetINTFactor 418
8－1－2. 基本関数詳細 420
8－1－2－1. MNET6045WReg 420
8－1－2－2. MNET6045RReg 422
8－1－2－3. MNET6045WCmd 424
8－1－2－4. MNET6045WPort 425
8－1－2－5. MNET6045RSts 426
8－2. 機能操作説明 428
8－2－1. 軸パターン 428
8－2－2. データの取得 429
8－2－2－1. 状態取得 429
8－2－2－2. カウンタ取得 431
8－2－2－3. 停止確認 431
8－2－2－4. エラー割込要因取得 431
8－2－2－5. イベント割込要因取得 431
8－2－3. スタートモード 432
8－2－4. 速度データ構造体 433

8－2－5. 移動データ構造体 437
8－2－6. 加減速設定 439
8－2－6－1. 加減速指定方法 439
8－2－6－2. 加減速設定の自動補正 439
8－2－7. S 字動作設定 440
8－2－8. 原点復帰モード 441
8－2－8－1. 原点復帰モード:0 441
8－2－8－2. 原点復帰モード:1 442
8－2－8－3. 原点復帰モード:2 442
8－2－8－4. 原点復帰モード:3 443
8－2－8－5. 原点復帰モード:4 443
8－2－8－6. 原点復帰モード:5 444
8－2－8－7. 原点復帰モード:6 444
8－2－8－8. 原点復帰モード:7 445
8－2－8－9. 原点復帰モード:8 445
8－2－8－10. 原点復帰モード:9 446
8－2－8－11. 原点復帰モード:10 446
8－2－8－12. 原点復帰モード:11 446
8－2－8－13. 原点復帰モード:12 447
8－2－9. 原点サーチ動作 448
8－2－10. 動作前確認 449
8－2－11. 速度設定に関する注意事項 450
8－2－11－1. 速度倍率 450
8－2－11－2. FH 補正 451
8－2－12. SD 入力 452
8－2－13. PCS 入力 452
8－2－14. 同期スタート 453
8－2－15. 同時停止 454
8－2－16. 基本固定パラメータ 454
8－2－17. コンパレータ 455
8－2－17－1. ソフトリミットの設定 458
8－2－18. イベント割込設定 460
8－2－19. ラッチ 461
8－2－20. 直線補間 462

8－2－21. 円弧補間 462
8－2－22. 次動作設定 463
8－2－23. 速度オーバーライド 463
8－2－24. 位置オーバーライド 464
8－2－25. パルサー同期動作 465
8－2－26. 操作の流れ 466
8－2－26－1. 操作フロー 466
8－2－26－2. プログラム記述例 467
8－3. データ定義 470
8－3－1. 構造体定義 470
8－3－1－1. ビットデータ構造体 470
8－3－2. ステータス／ビットマップ 474
8－3－2－1. エラー割込要因 474
8－3－2－2. イベント割込要因 475
8－3－3. 識別子 476
8－3－3－1. スタートモード 476
8－3－3－2. 動作方向 476
8－3－3－3. 加減速指定方法 476
8－3－3－4. S 字使用/未使用 476
8－3－3－5. カウンタ指定 476
8－3－3－6. ラッチ指定 477
8－3－3－7. 停止モード 477
8－3－3－8. 速度倍率レンジ 477
8－3－3－9. ビット選択 478
8－3－3－10. パラメータ項目番号定義 479
8－3－3－11. DLL 内部パラメータ項目番号定義 485
8－3－3－12. 戻り値 487
9. 関数インデックス 490
9－1. センターボード機種別関数 490
9－1－1. PPCI-L112 ボード関数 490
9－1－2. NPMCMNET-I/O104 ボード関数 490
9－1－3. MNET-PUSB3601 ボード関数 490
9－2. センターボード共通関数 490
9－3. ローカルボード共通関数 490

9－4. ローカルボード機種別関数 491
9－4－1. ローカル１軸モーションボード関数 491
9－4－2. ローカル４軸(6143)モーションボード関数 492
9－4－3. ローカル４軸(6045)モーションボード関数 493
10. 基本関数インデックス 495
10－1. センターデバイス基本関数 495
10－2. １軸モーションデバイス基本関数 495
10－3. ４軸(6045)モーションデバイス基本関数 495

1. はじめに

このたびは、超高速シリアル通信システム Motionnet シリーズのセンターボードをお求めいただきましてありがとうございます。

本取扱説明書は、Motionnet の DLL 関数について解説しております。

1−1. 概要

当 DLL には各種 Motionnet ローカル I/O、モーションボードに対応した関数が用意されております。この利用により Motionnet の Windows アプリケーション構築を容易に行うことができます。

1−2. 動作環境

対応するメインボードと DLL ファイル

メインボード	DLL ファイル
Motionnet センターボード(PCI) PPCI-L112	PPCI_L112.DLL(32bit 用) PPCI_L112_X64.DLL(64bit 用)
Motionnet センターボード(PC/104) NPMCMNET-I/O104	NPMCMNET104.DLL(32bit 用)
Motionnet センターユニット(USB) MNET-PUSB3601	MNETPUSB_X86.DLL(32bit 用) MNETPUSB_X64.DLL(64bit 用)

対応ＯＳ

```
Windows8 (32bit)      ... PPCI-L112,
Windows7 (32bit)      ... PPCI-L112,
WindowsVista (32bit) ... PPCI-L112,
WindowsXP             ... PPCI-L112, NPMCMNET-I/O104, MNET-PUSB3601
Windows2000           ... PPCI-L112, NPMCMNET-I/O104, MNET-PUSB3601

Windows8 (64bit)      ... PPCI-L112, MNET-PUSB3601
Windows7 (64bit)      ... PPCI-L112, MNET-PUSB3601
```

対応言語

```
Microsoft Visual C++
Microsoft Visual Basic
Microsoft Visual C#
```

1−3. DLL バージョン

本書に対応する DLL の製品バージョンは Ver2.00 です。

1−4. 導入

1−4−1. デバイスドライバのインストール

センターボードの取扱説明書を参照ください。

1−4−2. プロジェクトへのファイルの組み込み

1−4−2−1. PPCI_L112 の場合

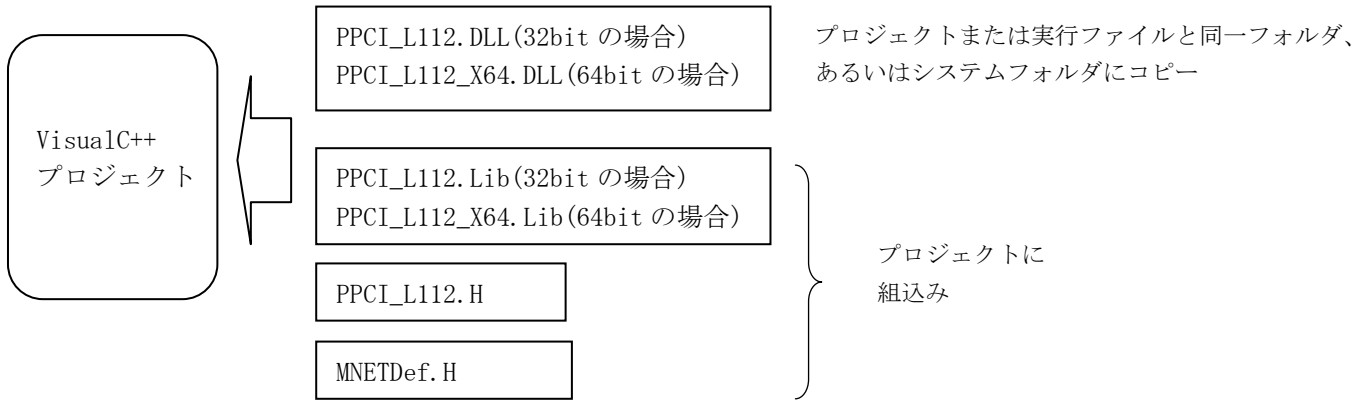
注意

（1） PPCI_L112.DLL(PPCI_L112_X64.DLL)はデバイスドライバインストール時にシステムフォルダにコピーされる PPCI-L112 ボードのローレベル DLL ”PCI_L112.DLL(PCI_L112_X64.DLL)” と ”MotionNET.DLL(Motionnet_X64.DLL)” の上位にリンクする形で構築されております。

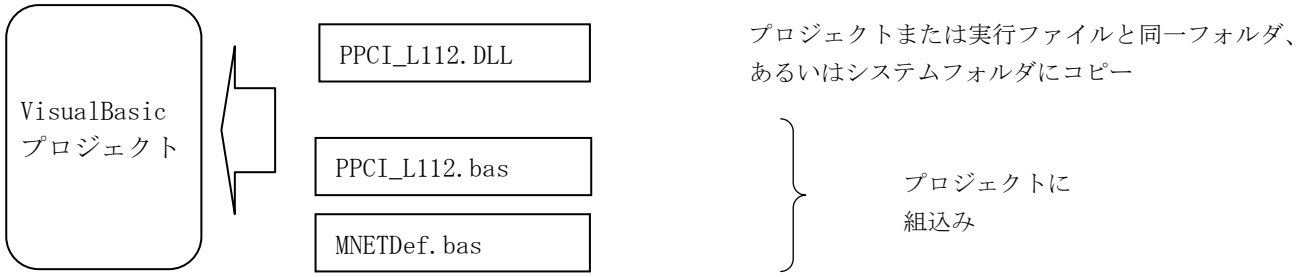
（2） デバイスドライバ、DLL、ローレベル DLL のファイルは同じパッケージ(CD-R など)で提供された組合せで御利用下さい。

（3） 上記の DLL は複数プロセスでの動作には対応しておりません。

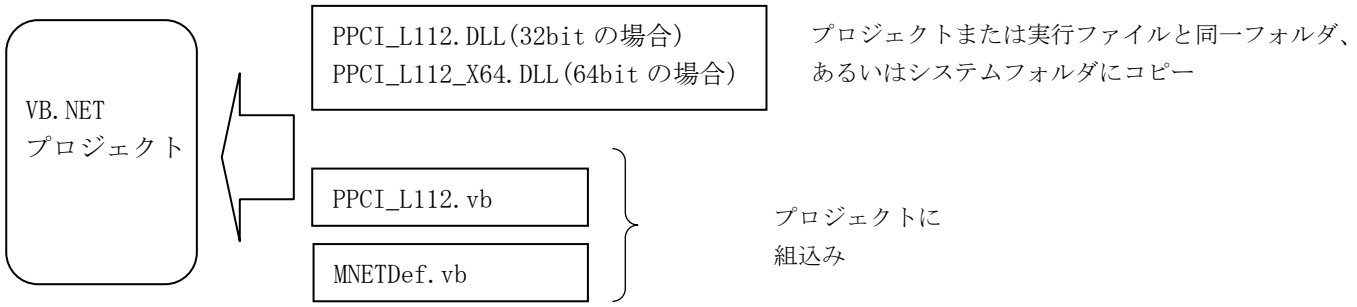
① VisualC++の場合



② VisualBasic の場合



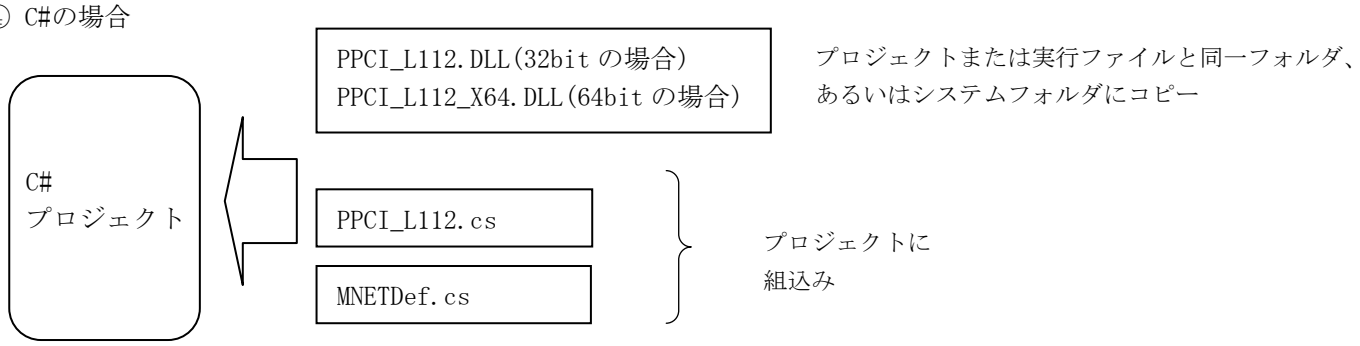
③ VB.NET の場合



注意

PPCI_L112.vbは32bit用/64bit用共ファイル名は同じですが、一部内容が異なりますのでご注意ください。
(各関数が参照する DLL ファイル名記述)

④ C#の場合



注意

PPCI_L112.csは32bit用/64bit用共ファイル名は同じですが、一部内容が異なりますのでご注意ください。
(各関数が参照する DLL ファイル名記述)

1－4－2－2. NPMCMNET-I/0104 の場合

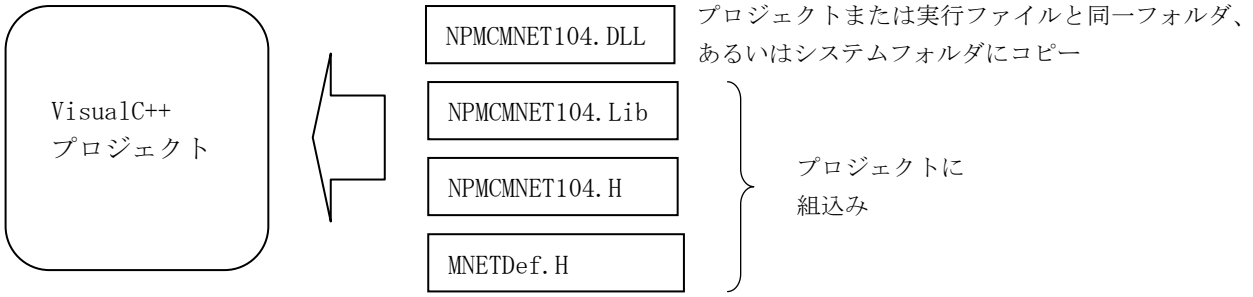
注意

（１） NPMCMNET104.DLL はデバイスドライバインストール時にシステムフォルダにコピーされる NPMCMNET-I/0104 ボードのローレベル DLL ”NPMCMNET104.dll” と ”MotionNET.DLL” の上位にリンクする形で構築されております。

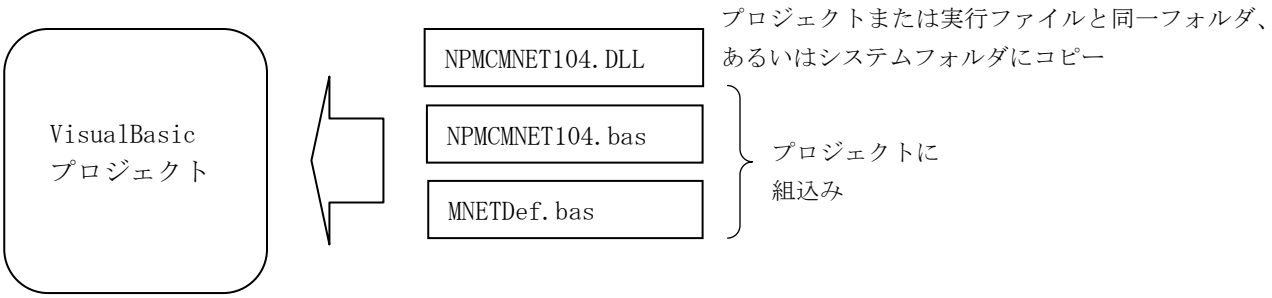
（２）デバイスドライバ、DLL、ローレベル DLL のファイルは同じパッケージ(CD-R など)で提供された組合せで御利用下さい。

（３）上記の DLL は複数プロセスでの動作には対応しておりません。

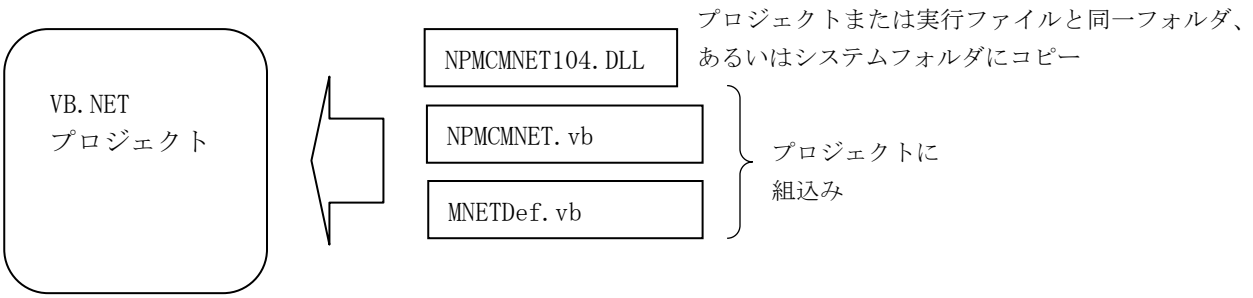
① VisualC++の場合



② VisualBasic の場合



③ VB.NET の場合



1－4－2－3. MNET-PUSB3601 の場合

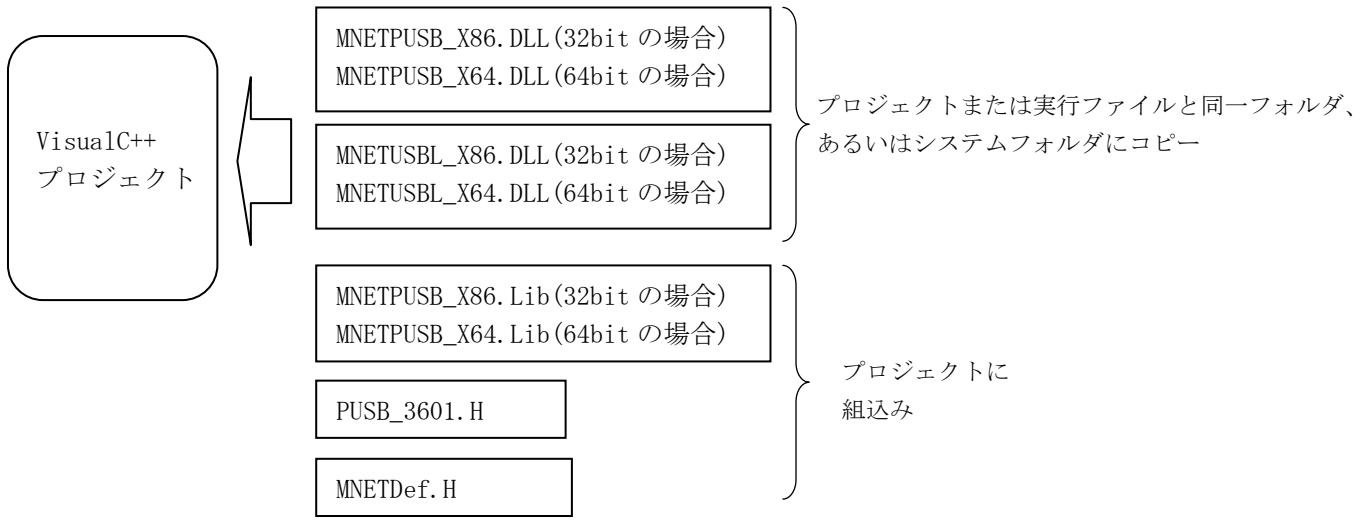
注意

（１） MNETPUSB_X86.DLL(MNETPUSB_X64.DLL)は MNET-PUSB3601 ボードのローレベル DLL ”MNETUSBL_X86.DLL(MNETUSBL_X64.DLL)”の上位にリンクする形で構築されております。

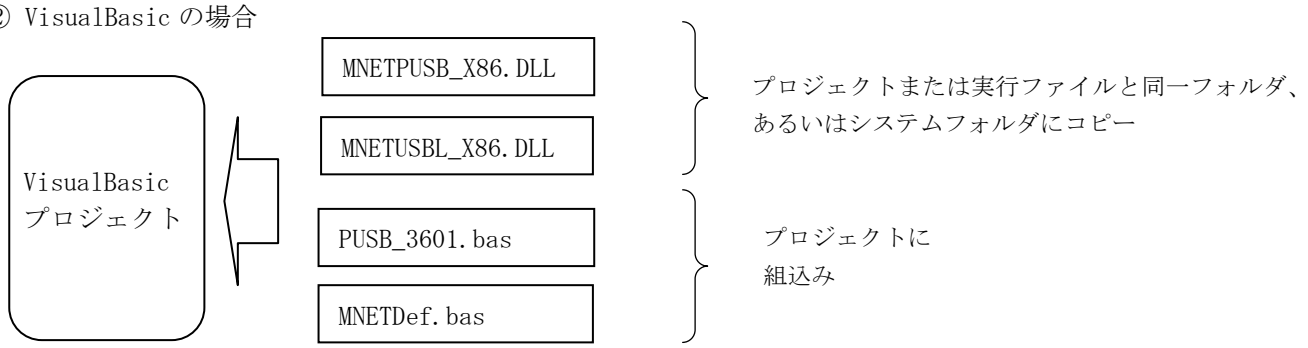
（２）デバイスドライバ、DLL、ローレベル DLL のファイルは同じパッケージ(CD-R など)で提供された組合せで御利用下さい。

（３）上記の DLL は複数プロセスでの動作には対応しておりません。

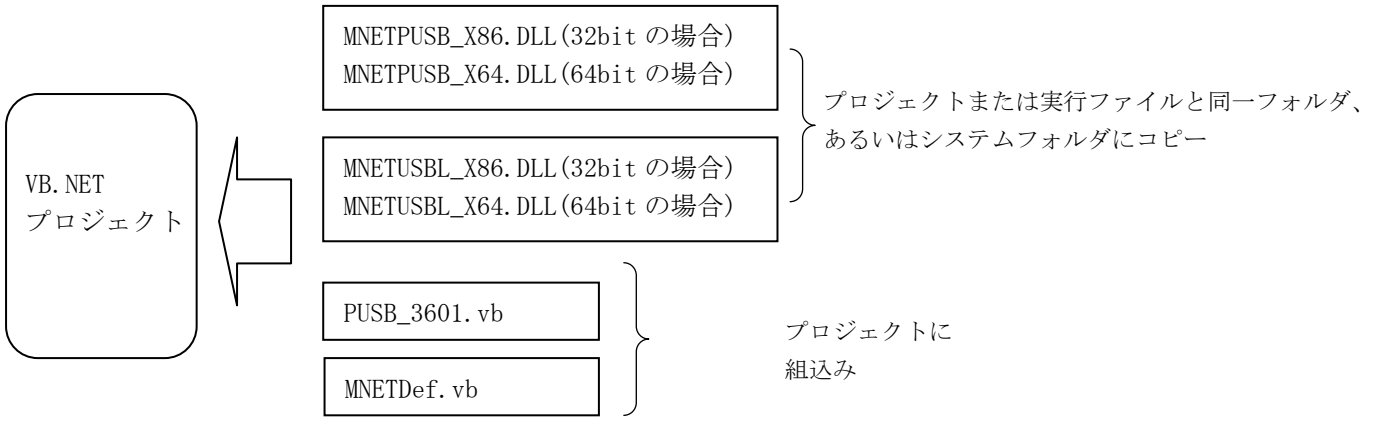
① VisualC++の場合



② VisualBasic の場合



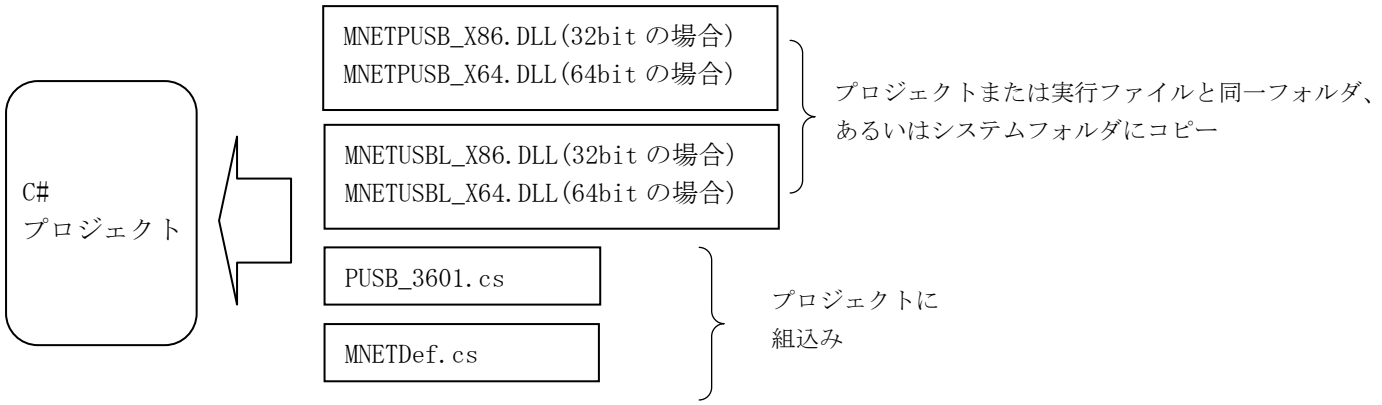
③ VB.NET の場合



注意

PUSB_3601.vbは32bit用/64bit用共ファイル名は同じですが、一部内容が異なりますのでご注意ください。
(各関数が参照する DLL ファイル名記述)

④ C#の場合



注意

PUSB_3601.csは32bit用/64bit用共ファイル名は同じですが、一部内容が異なりますのでご注意ください。
(各関数が参照する DLL ファイル名記述)

2. 機能構成

2−1. 機能分類

Motionnet の DLL 関数は次の様に分類されます、本取扱説明書もこの分類に則り構成されております。
ご利用にあたってはハードウェア構成に応じて該当する項目を参照ください。

(1) センターボード機種別機能
センターボードの機種別の機能です。センターボードのオープン/クローズ、またセンターボードに装備されたパラレル I/O を制御する機能などが含まれます。

(2) センターボード共通機能
センターボード全機種共通の機能です。 センターボードに搭載された Motionnet センターデバイス G9001A を制御する機能になります。

(3) ローカルボード共通機能
各種 Motionnet ローカルボード全機種共通の機能です。 ローカルボードに搭載された各種 Motionnet ローカルデバイスに共通するデータ(ポートデータ,デバイス情報,I/O 通信エラーフラグほか)を制御する機能になります。

(4) ローカルボード機種別機能
各種 Motionnet ローカルボード機種別の機能です。
次の機能などがここに分類されます。
・ローカル 1 軸モーションボード機能
・ローカル 4 軸(6143)モーションボード機能
・ローカル 4 軸(6045)モーションボード機能
（ローカル I/O ボードなどについては共通機能だけで制御できます）

注意
4 軸モーションボード関数対応基板型式
[ローカル 4 軸(6143)モーションボード関数]
・ MNET-BCDC5030A4
・ MNET-BCD4020FUA4
[ローカル 4 軸(6045)モーションボード関数]
・ MNET-M204-DUM

3. 関数一覧

3−1. センターボード機種別関数

3−1−1. PPCI-L112 ボード関数

No.	関数名称	機能
1	PPCI_L112Open	ボードオープン
2	PPCI_L112Close	ボードクローズ
3	PPCI_L112GetParaIn	パラレル入力読出し
4	PPCI_L112GetParaOut	パラレル出力読出し
5	PPCI_L112SetParaOut	パラレル出力書込み
6	PPCI_L112SetCommSpeed	通信速度設定

3−1−2. NPMCMNET-I/0104 ボード関数

No.	関数名称	機能
1	NPMCMNETOpen	ボードオープン
2	NPMCMNETClose	ボードクローズ
3	NPMCMNETGetParaIn	パラレル入力読出し
4	NPMCMNETGetParaOut	パラレル出力読出し
5	NPMCMNETSetParaOut	パラレル出力書込み
6	NPMCMNETSetCommSpeed	通信速度設定

3−1−3. MNET-PUSB3601 ボード関数

No.	関数名称	機能
1	MNETPUsbOpen	ボードオープン
2	MNETPUsbClose	ボードクローズ
3	MNETPUsbSetCommSpeed	通信速度設定

3−2. センターボード共通関数

No.	関数名称	機能
1	MNETSysComm	システム通信
2	MNETIOCommStart	I/O通信スタート
3	MNETIOCommStop	I/O通信ストップ
4	MNETCenterReset	センターデバイスリセット
5	MNETGetBRKF	ブレークフレーム有無確認
6	MNETGetEIOE	I/O通信エラー確認
7	MNETGetIOPC	入力ポート変化割込確認
8	MNETGetSBSY	I/O通信中確認
9	MNETGetDevType	ローカルデバイス種類読出し

3−3. ローカルボード共通関数

No.	関数名称	機能
1	MNETSetLocalData	ローカルデバイスデータ書込み
2	MNETGetLocalData	ローカルデバイスデータ読出し
3	MNETSetDLLPar	DLL内部パラメータ設定

3−4. ローカルボード機種別関数

3−4−1. ローカル 1 軸モーションボード関数

No.	関数名称	機能
1	MNETM1SetSpeed	速度設定
2	MNETM1Inc	Inc 移動
3	MNETM1Abs	Abs 移動
4	MNETM1Continue	連続移動
5	MNETM1Org	原点復帰移動
6	MNETM1EscOrg	原点脱出移動
7	MNETM1ToEL	EL まで移動
8	MNETM1EscEL	EL 脱出移動
9	MNETM1Ez	EZ カウント移動
10	MNETM1Stop	軸停止
11	MNETM1GetSpeed	軸速度取得
12	MNETM1GetCounter	カウンタ値取得
13	MNETM1SetCounter	カウンタ値設定
14	MNETM1GetLtc	ラッチデータ取得
15	MNETM1GetBitDataStruct	ビットデータ一括取得
16	MNETM1GetBitData	ビットデータ取得
17	MNETM1SetBitData	ビットデータ設定
18	MNETM1FixSpeedRange	速度倍率固定
19	MNETM1UnFixSpeedRange	速度倍率固定解除
20	MNETM1OutCsta	CSTA 信号出力
21	MNETM1OutCstp	CSTP 信号出力
22	MNETM1OvrideSpeed	速度変更
23	MNETM1OvridePos	目標位置オーバーライド
24	MNETM1ResetSEND	SEND(位置決め完了)ビットリセット
25	MNETM1SetPar	パラメータ設定
26	MNETM1GetPar	パラメータ取得
27	MNETM1SearchORG	原点サーチ移動
28	MNETM1GetREST	エラー割り込み要因取得
29	MNETM1GetRIST	イベント割り込み要因取得
30	MNETM1Restart	位置決めリスタート
31	MNETM1SoftPCS	PCS 入力代行コマンド発行
32	MNETM1SoftLTC	LTC 入力代行コマンド発行
33	MNETM1Reset	モーションデバイスリセット
34	MNETM1SetDLLPar	DLL 内部パラメータ設定
35	MNETM1ReadParFile	パラメータファイル読出し、デバイス設定
36	MNETM1WriteParFile	デバイスより取得し、パラメータファイル書込み

3−4−2. ローカル４軸(6143)モーションボード関数

No.	関数名称	機能
1	MNETM4SetSpeed	速度設定 (軸パターン指定)
2	MNETM4PtpInc	PtpInc移動 (軸パターン指定)
3	MNETM4LineInc	直線補間Inc移動 (軸パターン指定)
4	MNETM4Ptp	Ptp移動 (軸パターン指定)
5	MNETM4Line	直線補間移動 (軸パターン指定)
6	MNETM4Continue	連続移動 (軸パターン指定)
7	MNETM4Org	原点復帰移動 (軸パターン指定)
8	MNETM4SetSpeedNo	速度設定 (軸番号指定)
9	MNETM4PtpIncNo	PtpInc移動 (軸番号指定)
10	MNETM4PtpNo	Ptp移動 (軸番号指定)
11	MNETM4ContinueNo	連続移動 (軸番号指定)
12	MNETM4OrgNo	原点復帰移動 (軸番号指定)
13	MNETM4StopChk	軸停止確認
14	MNETM4GetREST	エラー割り込み要因取得
15	MNETM4GetRIST	イベント割り込み要因取得
16	MNETM4CancelPreReg	次動作キャンセル
17	MNETM4Stop	軸停止
18	MNETM4SetBitData	ビットデータ設定
19	MNETM4GetBitData	ビットデータ取得
20	MNETM4GetBitDataStruct	ビットデータ一括取得
21	MNETM4GetSpeed	軸速度取得
22	MNETM4GetCounter	カウンタ値取得
23	MNETM4SetCounter	カウンタ値設定
24	MNETM4GetLtc	ラッチデータ取得
25	MNETM4SoftLTC	LTC入力代行コマンド発行
26	MNETM4SoftPCS	PCS入力代行コマンド発行
27	MNETM4OutCsta	CSTA信号出力
28	MNETM4OutCstp	CSTP信号出力
29	MNETM4SetPar	パラメータ設定
30	MNETM4GetPar	パラメータ取得
31	MNETM4SetDLLPar	DLL内部パラメータ設定
32	MNETM4Reset	パルスコントロールLSIリセット
33	MNETM4ReadParFile	パラメータファイル読出し、デバイス設定
34	MNETM4WriteParFile	デバイスより取得し、パラメータファイル書込み

3−4−3. ローカル４軸(6045)モーションボード関数

No.	関数名称	機能
1	MNET6045SetSpeed	(非補間動作)速度設定 (軸パターン指定)
2	MNET6045PtpInc	Ptp移動(Inc指定) (軸パターン指定)
3	MNET6045Ptp	Ptp移動(Abs指定) (軸パターン指定)
4	MNET6045LineInc	直線補間移動(Inc指定) (軸パターン指定)
5	MNET6045Line	直線補間移動(Abs指定) (軸パターン指定)
6	MNET6045ArcInc	円弧補間移動(Inc指定) (軸パターン指定)
7	MNET6045Arc	円弧補間移動(Abs指定) (軸パターン指定)
8	MNET6045Continue	連続移動 (軸パターン指定)
9	MNET6045Org	原点復帰移動 (軸パターン指定)
10	MNET6045EscOrg	原点脱出移動 (軸パターン指定)
11	MNET6045SearchOrg	原点サーチ移動 (軸パターン指定)
12	MNET6045ToEL	ELまで移動 (軸パターン指定)
13	MNET6045EscEL	EL脱出移動 (軸パターン指定)
14	MNET6045Ez	EZカウント移動 (軸パターン指定)
15	MNET6045Stop	軸停止 (軸パターン指定)
16	MNET6045SetSpeedNo	(非補間動作)速度設定 (軸番号指定)
17	MNET6045PtpIncNo	Ptp移動(Inc指定) (軸番号指定)
18	MNET6045PtpNo	Ptp移動(Abs指定) (軸番号指定)
19	MNET6045ContinueNo	連続移動 (軸番号指定)
20	MNET6045OrgNo	原点復帰移動 (軸番号指定)
21	MNET6045EscOrgNo	原点脱出移動 (軸番号指定)
22	MNET6045SearchOrgNo	原点サーチ移動 (軸番号指定)
23	MNET6045ToELNo	ELまで移動 (軸番号指定)
24	MNET6045EscELNo	EL脱出移動 (軸番号指定)
25	MNET6045EzNo	EZカウント移動 (軸番号指定)
26	MNET6045StopNo	軸停止 (軸番号指定)
27	MNET6045GetBitStruct	ビットデータ一括取得
28	MNET6045GetBit	ビットデータ取得
29	MNET6045SetBit	ビットデータ設定
30	MNET6045GetSpeed	現在速度取得
31	MNET6045GetCounter	カウンタ値取得
32	MNET6045SetCounter	カウンタ値設定
33	MNET6045Reset	LSIリセット
34	MNET6045CancelPreReg	次動作キャンセル
35	MNET6045StopChk	軸停止確認
36	MNET6045SetDLLPar	DLLパラメータ設定
37	MNET6045SetPar	パラメータ設定
38	MNET6045GetPar	パラメータ取得
39	MNET6045FixSpeedRange	速度レンジ固定設定
40	MNET6045UnFixSpeedRange	速度レンジ固定設定解除
41	MNET6045SetFASpeed	補正速度設定
42	MNET6045OvrideSpd	速度オーバーライド
43	MNET6045GetRemain	移動残パルス取得
44	MNET6045GetLtc	ラッチデータ取得
45	MNET6045SoftLtc	ラッチ入力代行コマンド発行
46	MNET6045SoftPcs	PCS入力代行コマンド発行
47	MNET6045OutCsta	同時スタートコマンド発行
48	MNET6045OutCstp	同時ストップコマンド発行

No.	関数名称	機能
49	MNET6045GetTermAbs	最終到達位置データ取得
50	MNET6045OvridePos	目標位置オーバーライド
51	MNET6045ReadParFile	パラメータファイル読出し、デバイスに設定
52	MNET6045WriteParFile	デバイスより取得し、パラメータファイル書込み
53	MNET6045PulserPtpInc	パルサー同期Ptp移動(Inc指定)　(軸パターン指定)
54	MNET6045PulserPtp	パルサー同期Ptp移動(Abs指定)　(軸パターン指定)
55	MNET6045PulserLineInc	パルサー同期直線補間移動(Inc指定) (軸パターン指定)
56	MNET6045PulserLine	パルサー同期直線補間移動(Abs指定) (軸パターン指定)
57	MNET6045PulserArcInc	パルサー同期円弧補間移動(Inc指定) (軸パターン指定)
58	MNET6045PulserArc	パルサー同期円弧補間移動(Abs指定) (軸パターン指定)
59	MNET6045PulserContinue	パルサー同期連続移動　(軸パターン指定)
60	MNET6045PulserPtpIncNo	パルサー同期Ptp移動(Inc指定)　(軸番号指定)
61	MNET6045PulserPtpNo	パルサー同期Ptp移動(Abs指定)　(軸番号指定)
62	MNET6045PulserContinueNo	パルサー同期連続移動　(軸番号指定)
63	MNET6045GetINTFactor	割込要因取得

4. 基本関数

基本関数とはLSIへのレジスタ読書きなど、低レベルなアクセスを行なう関数です。
この低レベルな機能を使用されない場合はこの項の説明をお読み頂く必要はありません。
基本関数の説明にあたり、この項では基本関数以外の(普通の)関数を「複合関数」と呼ばせて頂きます。

4−1. 基本関数使用上の注意

基本関数と複合関数の混在は基本的にお避けください。
基本関数の使用にあたっては、LSIの取扱説明書などを十分に参照下さい。

4−1−1. センターデバイス関連

MNETGetCenterSts関数で取得するデータには読出しで消えるビットがあり、次の様に扱われております。
[BRKF] ：DLL内部に保持されます。MNETGetBRKF関数でDLL内部値を読み出す仕様になっております。
[CEND,EDTE,ERAE,CAER] ： レジスタ読出し、レジスタ書込み、コマンド書込みなどのDLL内部関数でチェックしております。 エラー関連のビットがONした場合は関数としてエラーを返す仕様となっておりますので、ユーザーレベルで確認を行う必要はありません。

4−1−2. ローカル1軸モーションデバイス関連

①複合関数は基本関数と同様のアクセスを複数回行って動作を実現しております。すべての関数には排他アクセス処理を組み込まれているため、一連のアクセスが割り込まれてしまうことはありませんが、複合関数と基本関数が混在する状況で、不用意にレジスタ値を変更された場合などは複合関数の動作に意図しない影響を及ぼす可能性があります。

②REST(エラー割り込み要因)レジスタ、RIST(イベント割り込み要因)レジスタなど読出しによりリセットされるデータがありますのでご注意ください。

4−1−3. ローカル4軸(6045)モーションデバイス関連

①複合関数は基本関数と同様のアクセスを複数回行って動作を実現しております。すべての関数には排他アクセス処理を組み込まれているため、一連のアクセスが割り込まれてしまうことはありませんが、複合関数と基本関数が混在する状況で、不用意にレジスタ値を変更された場合などは複合関数の動作に意図しない影響を及ぼす可能性があります。

②REST(エラー割り込み要因)レジスタ、RIST(イベント割り込み要因)レジスタなど読出しによりリセットされるデータがありますのでご注意ください。

4−2. 基本関数一覧

4−2−1. センターデバイス基本関数

No.	関数名称	機能
1	MNETSetCenterCmd	センターデバイスコマンド書込み
2	MNETGetCenterSts	センターデバイスステータス読出し
3	MNETGetCenterIntSts	センターデバイス割込ステータス読出し
4	MNETWriteCenterRegister	センターデバイスレジスタ書込み
5	MNETReadCenterRegister	センターデバイスレジスタ読出し

4−2−2. 1軸モーションデバイス基本関数

No.	関数名称	機能
1	MNETM1WriteRegister	モーションデバイスレジスタ書込み
2	MNETM1ReadRegister	モーションデバイスレジスタ読出し
3	MNETM1WriteCommand	モーションデバイスコマンド書込み

4−2−3. 4軸(6045)モーションデバイス基本関数

No.	関数名称	機能
1	MNET6045WReg	モーションデバイスレジスタ書込み
2	MNET6045RReg	モーションデバイスレジスタ読出し
3	MNET6045WCmd	モーションデバイスコマンド書込み
4	MNET6045WPort	モーションデバイスポート書込み
5	MNET6045RSts	モーションデバイスステータス読出し

5. センターボードおよび、ローカルボード共通機能

5−1. 関数詳細

関数詳細説明にあたって

(1) 本 DLL が対応している最大センターボード数は 4 です。(MNET-PUSB3601 については 1 です。)

(2) 本 DLL は割り込み機能は有しておりません。

以降の説明に「割り込み」という言葉を用いておりますが、これはセンターボードおよびローカルボード上の LSI(G9001A,G9002,G9003,G9004A,PCL6143,PCL6045)に装備された割り込み機能を示しております。

具体的には本 DLL は LSI が発生している INT 信号の処理は行っていませんが、LSI の割り込み状態を示すレジスタやビットは INT 信号を使う場合と同様に動作し、ソフト側からモニタすることが可能です。

したがって「割り込みが立つ」、「割り込みが発生する」などの表現は、「割り込み状態を示すステータスやビットが ON する」とご理解ください。

(3) 各関数引数内の、(パラレル入出力)ポート番号、ライン番号、デバイス番号などの初期値(1 個目を示す値)は 0 です。

以降の説明では混乱を避けるため、関数で指定する値(0 始まり)で説明しております。

(例) 1 ライン目の 2 個目のデバイス -> ライン 0 のデバイス 1

(4) (パラレル入出力)ポート番号について

1 枚目のボードの入力ポート番号、出力ポート番号が 0。 2 枚目のボードの入力ポート番号、出力ポート番号が 1 となります。

(5) ライン番号について

1 枚目のボードの 1 ライン目がライン番号:0、2 ライン目がライン番号:1。 2 枚目のボードの 1 ライン目がライン番号:2、2 ライン目がライン番号:3 となります。

MNET-PUSB3601 については最大センターボード数が 1 で、また MNET-PUSB3601 が持つライン数は 1 ですので最大ライン番号は 0 となります。

(6) VB 書式の記述について

センターボード機種別関数以外の VB 書式説明記述において、Lib 節で指定している DLL の名称については、便宜上"PPCI_L112.DLL"としております。

5－1－1. PPCI-L112 ボード関数

5－1－1－1. PPCI_L112Open

機能

ボードオープン
システム上の全てのPPCI-L112ボードがオープンされ、アクセスが可能となります。

書式

[VisualC++]
```
short __stdcall PPCI_L112Open(void);
```
[VisualBasic]
```
Declare Function PPCI_L112Open Lib "PPCI_L112.dll" () As Integer
```
[VB.NET]
```
Declare Function PPCI_L112Open Lib "PPCI_L112.dll" () As Short
```

パラメータ

なし。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 参照）

使用例

[VisualC++]
```
short result;                       // 結果

result = PPCI_L112Open();           // ボードオープン
```

[VisualBasic]
```
Dim result As Integer               ' 結果

result = PPCI_L112Open()            ' ボードオープン
```

[VB.NET]
```
Dim result As Short                 ' 結果

result = PPCI_L112Open()            ' ボードオープン
```

以上によりPPCI-L112ボードがオープンされ、アクセスが可能となります。

5－1－1－2. PPCI_L112Close

機能

ボードクローズ
システム上の全てのPPCI-L112ボードをクローズします。

書式

[VisualC++]
```
short __stdcall PPCI_L112Close(void);
```
[VisualBasic]
```
Declare Function PPCI_L112Close lib "PPCI_L112.dll" () As Integer
```
[VB.NET]
```
Declare Function PPCI_L112Close lib "PPCI_L112.dll" () As Short
```

パラメータ

なし。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 参照）

使用例

[VisualC++]
```
short result;                          // 結果

result = PPCI_L112Close();             // ボードクローズ
```
[VisualBasic]
```
Dim result As Integer                  ’結果

result = PPCI_L112 Close()             ’ボードクローズ
```
[VB.NET]
```
Dim result As Short                    ’結果

result = PPCI_L112 Close()             ’ボードクローズ
```

以上によりPPCI-L112ボードがクローズされます。

5－1－1－3. PPCI_L112GetParaIn

機能

パラレル入力読出し

書式

[VisualC++]

```
short __stdcall PPCI_L112GetParaIn(
      short portNo,
      unsigned char* data);
```

[VisualBasic]

```
Declare Function PPCI_L112GetParaIn Lib "PPCI_L112.dll" (
      ByVal portNo As Integer,
      ByRef data As Byte) As Integer
```

[VB.NET]

```
Declare Function PPCI_L112GetParaIn Lib "PPCI_L112.dll" (
      ByVal portNo As Short,
      ByRef data As Byte) As Short
```

パラメータ

portNo	***入力ポート番号***（0始まり）
data	***取得データ格納変数***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 参照）

使用例

[VisualC++]

```
char data;

PPCI_L112GetParaIn(0, &data);
```

[VisualBasic]

```
Dim data As Byte

PPCI_L112GetParaIn 0, data
```

[VB.NET]

```
Dim data As Byte

PPCI_L112GetParaIn(0, data)
```

入力ポート0のパラレル入力ポート状態を変数dataに取得します。

5－1－1－4. PPCI_L112GetParaOut

機能

パラレル出力読出し

書式

[VisualC++]

```
short __stdcall PPCI_L112GetParaOut(
      short portNo,
      unsigned char* data);
```

[VisualBasic]

```
Declare Function PPCI_L112GetParaOut Lib "PPCI_L112.dll" (
      ByVal portNo As Integer,
      ByRef data As Byte) As Integer
```

[VB.NET]

```
Declare Function PPCI_L112GetParaOut Lib "PPCI_L112.dll" (
      ByVal portNo As Short,
      ByRef data As Byte) As Short
```

パラメータ

portNo	***出力ポート番号***（0始まり）
data	***取得データ格納変数***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 参照）

使用例

[VisualC++]

```
char data;

PPCI_L112GetParaOut(0, &data);
```

[VisualBasic]

```
Dim data As Byte

PPCI_L112GetParaOut 0, data
```

[VB.NET]

```
Dim data As Byte

PPCI_L112GetParaOut(0, data)
```

出力ポート 0 の状態を変数 data に取得します。

5－1－1－5. PPCI_L112SetParaOut

機能

パラレル出力書込み

書式

[VisualC++]

```
short __stdcall PPCI_L112SetParaOut(
      short portNo,
      unsigned char data);
```

[VisualBasic]

```
Declare Function PPCI_L112SetParaOut Lib "PPCI_L112.dll" (
      ByVal portNo As Integer,
      ByVal data As Byte) As Integer
```

[VB.NET]

```
Declare Function PPCI_L112SetParaOut Lib "PPCI_L112.dll" (
      ByVal portNo As Short,
      ByVal data As Byte) As Short
```

パラメータ

portNo ***出力ポート番号*** (0始まり)

data ***設定値***

設定可能範囲は0～Fhとなります。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 参照)

使用例

[VisualC++]

```
PPCI_L112GetParaOut(0, 0x2);
```

[VisualBasic]

```
PPCI_L112GetParaOut 0, &H2
```

[VB.NET]

```
PPCI_L112GetParaOut(0, &H2)
```

出力ポート0にデータ2h(2Bit目だけON)を設定します。

5－1－1－6. PPCI_L112SetCommSpeed

機能

通信速度設定

書式

[VisualC++]

```
short __stdcall PPCI_L112SetCommSpeed(
      short lineNo,
      unsigned short commSpeed);
```

[VisualBasic]

```
Declare Function PPCI_L112SetCommSpeed Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal commSpeed As Integer) As Integer
```

[VB.NET]

```
Declare Function PPCI_L112SetCommSpeed Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal commSpeed As Short) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

commSpeed ***通信速度識別子***

識別子	値	内容
MNET_COMMSPEED_20M	3	20Mbps
MNET_COMMSPEED_10M	2	10Mbps
MNET_COMMSPEED_5M	1	5Mbps
MNET_COMMSPEED_2_5M	0	2.5Mbps

注意

・通信速度設定はボードオープン後、すべてのMotionnetの処理(システム通信ほか)を行う前に行ってください。
・通信速度設定後にセンターデバイスのリセットを行ってください。これにより設定した通信速度が適用されます。
・ローカルデバイスの通信速度設定と同じに設定する必要があります。

備考

当関数を実行しない場合は20Mbps設定となります。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 参照)

使用例

[VisualC++]

```
PPCI_L112SetCommSpeed(1, MNET_COMMSPEED_10M);    //通信速度設定
MNETCenterReset(1);                              //センターデバイスリセット
```

[VisualBasic]

```
PPCI_L112SetCommSpeed 1, MNET_COMMSPEED_10M    ’通信速度設定
MNETCenterReset 1                              ’センターデバイスリセット
```

[VB.NET]

```
PPCI_L112SetCommSpeed(1, MNET_COMMSPEED_10M)    ’通信速度設定
MNETCenterReset( 1 )                            ’センターデバイスリセット
```

ライン1の通信速度を10Mbpsに設定します。

5－1－2. NPMCMNET-I/0104 ボード関数

5－1－2－1. NPMCMNETOpen

機能

ボードオープン
システム上の全てのNPMCMNET-I/0104ボードがオープンされ、アクセスが可能となります。

書式

[VisualC++]

```
short __stdcall NPMCMNETOpen(void);
```

[VisualBasic]

```
Declare Function NPMCMNETOpen Lib "NPMCMNET104.dll" () As Integer
```

[VB.NET]

```
Declare Function NPMCMNETOpen Lib "NPMCMNET104.dll" () As Short
```

パラメータ

なし。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 参照）

使用例

[VisualC++]

```
short result;                   // 結果

result = NPMCMNETOpen();        // ボードオープン
```

[VisualBasic]

```
Dim result As Integer           ’結果

result = NPMCMNETOpen()         ’ボードオープン
```

[VB.NET]

```
Dim result As Short             ’結果

result = NPMCMNETOpen()         ’ボードオープン
```

以上によりNPMCMNET-I/0104ボードがオープンされ、アクセスが可能となります。

5－1－2－2. NPMCMNETClose

機能

ボードクローズ

システム上の全てのNPMCMNET-I/0104ボードをクローズします。

書式

[VisualC++]

```
short __stdcall NPMCMNETClose(void);
```

[VisualBasic]

```
Declare Function NPMCMNETClose lib " NPMCMNET104.dll" () As Integer
```

[VB.NET]

```
Declare Function NPMCMNETClose lib " NPMCMNET104.dll" () As Short
```

パラメータ

なし。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 参照）

使用例

[VisualC++]

```
short result;                   // 結果

result = NPMCMNETClose();       // ボードクローズ
```

[VisualBasic]

```
Dim result As Integer           ’結果

result = NPMCMNETClose()        ’ボードクローズ
```

[VB.NET]

```
Dim result As Short             ’結果

result = NPMCMNETClose()        ’ボードクローズ
```

以上によりNPMCMNET-I/0104ボードがクローズされます。

5－1－2－3. NPMCMNETGetParaIn

機能

パラレル入力読出し

書式

[VisualC++]

```
short__stdcall NPMCMNETGetParaIn(
      short portNo,
      unsigned char* data);
```

[VisualBasic]

```
Declare Function NPMCMNETGetParaIn Lib "NPMCMNET104.dll" (
      ByVal portNo As Integer,
      ByRef data As Byte) As Integer
```

[VB.NET]

```
Declare Function NPMCMNETGetParaIn Lib "NPMCMNET104.dll" (
      ByVal portNo As Short,
      ByRef data As Byte) As Short
```

パラメータ

portNo	***入力ポート番号***（0始まり）
data	***取得データ格納変数***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 参照）

使用例

[VisualC++]

```
char data;

NPMCMNETGetParaIn(0, &data);
```

[VisualBasic]

```
Dim data As Byte

NPMCMNETGetParaIn 0, data
```

[VB.NET]

```
Dim data As Byte

NPMCMNETGetParaIn(0, data)
```

入力ポート0のパラレル入力ポート状態を変数dataに取得します。

5－1－2－4. NPMCMNETGetParaOut

機能

パラレル出力読出し

書式

[VisualC++]

```
short __stdcall NPMCMNETGetParaOut(
      short portNo,
      unsigned char* data);
```

[VisualBasic]

```
Declare Function NPMCMNETGetParaOut Lib "NPMCMNET104.dll" (
      ByVal portNo As Integer,
      ByRef data As Byte) As Integer
```

[VB.NET]

```
Declare Function NPMCMNETGetParaOut Lib "NPMCMNET104.dll" (
      ByVal portNo As Short,
      ByRef data As Byte) As Short
```

パラメータ

portNo	***出力ポート番号***（0始まり）
data	***取得データ格納変数***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 参照）

使用例

[VisualC++]

```
char data;

NPMCMNETGetParaOut(0, &data);
```

[VisualBasic]

```
Dim data As Byte

NPMCMNETGetParaOut 0, data
```

[VB.NET]

```
Dim data As Byte

NPMCMNETGetParaOut(0, data)
```

出力ポート0の状態を変数dataに取得します。

5－1－2－5. NPMCMNETSetParaOut

機能

パラレル出力書込み

書式

[VisualC++]

```
short __stdcall NPMCMNETSetParaOut(
      short portNo,
      unsigned char data);
```

[VisualBasic]

```
Declare Function NPMCMNETSetParaOut Lib "NPMCMNET104.dll" (
      ByVal portNo As Integer,
      ByVal data As Byte) As Integer
```

[VB.NET]

```
Declare Function NPMCMNETSetParaOut Lib "NPMCMNET104.dll" (
      ByVal portNo As Short,
      ByVal data As Byte) As Short
```

パラメータ

portNo ***出力ポート番号*** (0始まり)

data ***設定値***

設定可能範囲は0～FFhとなります。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 参照)

使用例

[VisualC++]

```
NPMCMNETGetParaOut(0, 0x2);
```

[VisualBasic]

```
NPMCMNETGetParaOut 0, &H2
```

[VB.NET]

```
NPMCMNETGetParaOut(0, &H2)
```

出力ポート0にデータ2h(2Bit目だけON)を設定します。

5－1－2－6. NPMCMNETSetCommSpeed

機能

通信速度設定

書式

[VisualC++]

```
short __stdcall NPMCMNETSetCommSpeed(
      short lineNo,
      unsigned short commSpeed);
```

[VisualBasic]

```
Declare Function NPMCMNETSetCommSpeed Lib "NPMCMNET104.dll" (
      ByVal lineNo As Integer,
      ByVal commSpeed As Integer) As Integer
```

[VB.NET]

```
Declare Function NPMCMNETSetCommSpeed Lib "NPMCMNET104.dll" (
      ByVal lineNo As Short,
      ByVal commSpeed As Integer) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

commSpeed ***通信速度識別子***

識別子	値	内容
MNET_COMMSPEED_20M	3	20Mbps
MNET_COMMSPEED_10M	2	10Mbps
MNET_COMMSPEED_5M	1	5Mbps
MNET_COMMSPEED_2_5M	0	2.5Mbps

注意

・通信速度設定はボードオープン後、すべてのMotionnetの処理(システム通信ほか)を行う前に行ってください。
・通信速度設定後にセンターデバイスのリセットを行ってください。これにより設定した通信速度が適用されます。
・ローカルデバイスの通信速度設定と同じに設定する必要があります。

備考

当関数を実行しない場合は20Mbps設定となります。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 参照)

使用例

[VisualC++]

```
NPMCMNETSetCommSpeed(1, MNET_COMMSPEED_10M);      //通信速度設定
MNETCenterReset(1);                               //センターデバイスリセット
```

[VisualBasic]

```
NPMCMNETSetCommSpeed 1, MNET_COMMSPEED_10M        ’通信速度設定
MNETCenterReset 1                                 ’センターデバイスリセット
```

[VB.NET]

```
NPMCMNETSetCommSpeed(1, MNET_COMMSPEED_10M)       ’通信速度設定
MNETCenterReset(1)                                ’センターデバイスリセット
```

ライン1の通信速度を10Mbpsに設定します。

5－1－3. MNET-PUSB3601 ボード関数

5－1－3－1. MNETPUsbOpen

機能

ボードオープン

MNET-PUSB3601ボードがオープンされ、アクセスが可能となります。

書式

[VisualC++]

```
short __stdcall MNETPUsbOpen(void);
```

[VisualBasic]

```
Declare Function MNETPUsbOpen Lib "MNETPUSB.dll" () As Integer
```

[VB.NET]

```
Declare Function MNETPUsbOpen Lib "MNETPUSB.dll" () As Short
```

パラメータ

なし。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 参照）

使用例

[VisualC++]

```
short result;                       // 結果

result = MNETPUsbOpen();            // ボードオープン
```

[VisualBasic]

```
Dim result As Integer               ' 結果

result = MNETPUsbOpen()             ' ボードオープン
```

[VB.NET]

```
Dim result As Short                 ' 結果

result = MNETPUsbOpen()             ' ボードオープン
```

以上によりMNET-PUSB3601ボードがオープンされ、アクセスが可能となります。

5－1－3－2. MNETPUsbClose

機能

ボードクローズ

MNET-PUSB3601ボードをクローズします。

書式

[VisualC++]

```
short __stdcall MNETPUsbClose(void);
```

[VisualBasic]

```
Declare Function MNETPUsbClose lib " MNETPUSB.dll" () As Integer
```

[VB.NET]

```
Declare Function MNETPUsbClose lib " MNETPUSB.dll" () As Short
```

パラメータ

なし。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 参照）

使用例

[VisualC++]

```
short result;                          // 結果

result = MNETPUsbClose();              // ボードクローズ
```

[VisualBasic]

```
Dim result As Integer                  ' 結果

result = MNETPUsbClose()               ' ボードクローズ
```

[VB.NET]

```
Dim result As Short                    ' 結果

result = MNETPUsbClose()               ' ボードクローズ
```

以上によりMNET-PUSB3601ボードがクローズされます。

5－1－3－3. MNETPUsbSetCommSpeed

機能

通信速度設定

書式

[VisualC++]

```
short __stdcall MNETPUsbSetCommSpeed(
      short lineNo,
      unsigned short commSpeed);
```

[VisualBasic]

```
Declare Function MNETPUsbSetCommSpeed Lib " MNETPUSB.dll" (
      ByVal lineNo As Integer,
      ByVal commSpeed As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETPUsbSetCommSpeed Lib " MNETPUSB.dll" (
      ByVal lineNo As Short,
      ByVal commSpeed As Short) As Integer
```

パラメータ

lineNo ***ライン番号*** (0始まり)

commSpeed ***通信速度識別子***

識別子	値	内容
MNET_COMMSPEED_20M	3	20Mbps
MNET_COMMSPEED_10M	2	10Mbps
MNET_COMMSPEED_5M	1	5Mbps
MNET_COMMSPEED_2_5M	0	2.5Mbps

注意

・通信速度設定はボードオープン後、すべてのMotionnetの処理(システム通信ほか)を行う前に行ってください。
・通信速度設定後にセンターデバイスのリセットを行ってください。これにより設定した通信速度が適用されます。
・ローカルデバイスの通信速度設定と同じに設定する必要があります。

備考

当関数を実行しない場合は20Mbps設定となります。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 参照)

使用例

[VisualC++]

```
MNETPUsbSetCommSpeed(0, MNET_COMMSPEED_10M);      //通信速度設定
MNETCenterReset(0);                               //センターデバイスリセット
```

[VisualBasic]

```
MNETPUsbSetCommSpeed 0, MNET_COMMSPEED_10M        ’通信速度設定
MNETCenterReset 0                                 ’センターデバイスリセット
```

[VB.NET]

```
MNETPUsbSetCommSpeed(0, MNET_COMMSPEED_10M)       ’通信速度設定
MNETCenterReset(0)                                ’センターデバイスリセット
```

ライン0の通信速度を10Mbpsに設定します。

5－1－4. センターボード共通関数

5－1－4－1. MNETSysComm

機能

システム通信

書式

[VisualC++]

```
short __stdcall MNETSysComm(
      short lineNo);
```

[VisualBasic]

```
Declare Function MNETSysComm Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETSysComm Lib "PPCI_L112.dll" (
      ByVal lineNo As Short) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 参照)

備考

この関数はシステム通信コマンド発行後、システム通信完了で返ります。

使用例

[VisualC++]

```
MNETSysComm(0);
```

[VisualBasic]

```
MNETSysComm 0
```

[VB.NET]

```
MNETSysComm(0)
```

ライン0のシステム通信を行います。

5－1－4－2. MNETIOCommStart

機能

I/O通信スタート

書式

[VisualC++]

```
short __stdcall MNETIOCommStart(
      short lineNo);
```

[VisualBasic]

```
Declare Function MNETIOCommStart Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETIOCommStart Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 参照）

備考

この関数はI/O通信スタートコマンド発行、I/O通信スタート後データ通信を行い、ライン上のローカルデバイスの属性をDLL内部に取得します。

I/O通信がスタートできない場合は「I/O通信開始タイムアウトエラー」となります。

使用例

[VisualC++]

```
MNETIOCommStart(0);
```

[VisualBasic]

```
MNETIOCommStart 0
```

[VB.NET]

```
MNETIOCommStart(0)
```

ラインのI/O通信をスタートします。

5－1－4－3. MNETIOCommStop

機能

I/O通信ストップ

書式

[VisualC++]

```
short __stdcall MNETIOCommStop(
      short lineNo);
```

[VisualBasic]

```
Declare Function MNETIOCommStop Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETIOCommStop Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 参照)

備考

この関数はI/O通信ストップコマンド発行後、I/O通信ストップ完了で返ります。

使用例

[VisualC++]

```
MNETIOCommStop(0);
```

[VisualBasic]

```
MNETIOCommStop 0
```

[VB.NET]

```
MNETIOCommStop(0)
```

ライン0のI/O通信をストップします。

5－1－4－4. MNETCenterReset

機能

センターデバイスリセット

書式

[VisualC++]

```
short __stdcall MNETCenterReset(
      short lineNo);
```

[VisualBasic]

```
Declare Function MNETCenterReset Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETCenterReset Lib "PPCI_L112.dll" (
      ByVal lineNo As Short) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 参照）

備考

この関数はセンターデバイスリセットコマンド発行後、リセット処理完了で返ります。

使用例

[VisualC++]

```
MNETCenterReset(0);
```

[VisualBasic]

```
MNETCenterReset 0
```

[VB.NET]

```
MNETCenterReset(0)
```

ライン0のセンターデバイスをリセットします。

5－1－4－5. MNETGetBRKF

機能

ブレークフレーム有無確認

書式

[VisualC++]

```
short __stdcall MNETGetBRKF(
      short lineNo,
      char* brkf);
```

[VisualBasic]

```
Declare Function MNETGetBRKF Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByRef brkf As Byte) As Integer
```

[VB.NET]

```
Declare Function MNETGetBRKF Lib "PPCI_L112.dll" (
      ByVal lineNo As Short
      ByRef brkf As Byte) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

brkf ***取得データ格納変数***
ブレークフレーム割り込みがあれば 1, なければ 0 が取得されます。
状態は読出しによって 0 にリセットされます。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 参照）

使用例

[VisualC++]

```
char brkf;

MNETGetBRKF (0, &brkf);
```

[VisualBasic]

```
Dim brkf as Byte

MNETGetBRKF 0, brkf
```

[VB.NET]

```
Dim brkf as Byte

MNETGetBRKF(0, brkf)
```

ライン0のブレークフレームの有無を変数brkfに取得します。

5－1－4－6. MNETGetEIOE

機能

I/O通信エラー有無確認

書式

[VisualC++]

```
short __stdcall MNETGetEIOE(
      short lineNo,
      char* eioe);
```

[VisualBasic]

```
Declare Function MNETGetEIOELib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByRef eioe As Byte) As Integer
```

[VB.NET]

```
Declare Function MNETGetEIOELib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByRef eioe As Byte) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

eioe ***取得データ格納変数***

I/O 通信エラーが発生していれば 1, していなければ 0 が取得されます。
この状態は指定されたラインに接続されたローカルデバイスの I/O 通信エラーフラグの OR となります。
各ローカルデバイスの状態の確認、および状態のクリアは MNETGetLocalData 関数, MNETSetLocalData 関数を使用します。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 参照)

使用例

[VisualC++]

```
char eioe;

MNETGetEIOE (0, &eioe);
```

[VisualBasic]

```
Dim eioe as Byte

MNETGetEIOE 0, eioe
```

[VB.NET]

```
Dim eioe as Byte

MNETGetEIOE(0, eioe)
```

ライン0のI/O通信エラーの有無を変数eioeに取得します。

Motionnet® Remote I/O & Remote Motion

5－1－4－7. MNETGetIOPC

機能

入力ポート変化割込有無確認

書式

[VisualC++]

```
short __stdcall MNETGetIOPC(
      short lineNo,
      char* iopc);
```

[VisualBasic]

```
Declare Function MNETGetIOPC Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByRef iopc As Byte) As Integer
```

[VB.NET]

```
Declare Function MNETGetIOPC Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByRef iopc As Byte) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

iopc ***取得データ格納変数***

入力ポート変化割込が発生していれば 1, していなければ 0 が取得されます。
この状態は指定されたラインに接続されたローカルデバイスの入力ポート変化割込フラグの OR となります。各ローカルデバイスの状態の確認、および状態のクリアは MNETGetLocalData 関数,MNETSetLocalData 関数を使用します。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 参照）

使用例

[VisualC++]

```
char iopc;

MNETGetIOPC (0, &iopc);
```

[VisualBasic]

```
Dim iopc as Byte

MNETGetIOPC 0, iopc
```

[VB.NET]

```
Dim iopc as Byte

MNETGetIOPC(0, iopc)
```

ライン0の入力ポート変化割込の有無を変数iopcに取得します。

5－1－4－8. MNETGetSBSY

機能

I/O通信中確認

書式

[VisualC++]

```
short __stdcall MNETGetSBSY(
      short lineNo,
      char* sbsy);
```

[VisualBasic]

```
Declare Function MNETGetSBSY Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByRef sbsy As Byte) As Integer
```

[VB.NET]

```
Declare Function MNETGetSBSY Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByRef sbsy As Byte) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

sbsy ***取得データ格納変数***

I/O 通信が行われていれば 1, 行われていなけしていなければ 0 が取得されます。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 参照)

使用例

[VisualC++]

```
char sbsy;

MNETGetSBSY (0, &sbsy);
```

[VisualBasic]

```
Dim sbsy as Byte

MNETGetSBSY 0, sbsy
```

[VB.NET]

```
Dim sbsy as Byte

MNETGetSBSY(0, sbsy)
```

ライン0のI/O通信が行われているかどうかを変数sbsyに取得します。

5－1－4－9. MNETGetDevType

機能

ローカルデバイス種類読出し

書式

[VisualC++]

```
short __stdcall MNETGetDevType(
      short lineNo,
      short devNo,
      short* typ);
```

[VisualBasic]

```
Declare Function MNETGetDevType Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal lineNo As Integer,
      ByRef typ As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETGetDevType Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal lineNo As Short,
      ByRef typ As Short) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
typ	***種類データ格納変数***
	2 : I/Oデバイス
	3 : 1軸モーションデバイス
	4 : 4軸モーションデバイス

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 参照)

使用例

[VisualC++]

```
short typ;

MNETGetDevType (0, 3, &typ);
```

[VisualBasic]

```
Dim typ as Integer

MNETGetDevType 0, 3, typ
```

[VB.NET]

```
Dim typ as Short

MNETGetDevType(0, 3, typ)
```

ライン0, デバイス3の種類を示すデータを取得します。

5－1－5. センターデバイス基本関数

5－1－5－1. MNETSetCenterCmd

機能

センターデバイスコマンド書込み

書式

[VisualC++]

```
short __stdcall MNETSetCenterCmd(
      short lineNo,
      short cmd);
```

[VisualBasic]

```
Declare Function MNETSetCenterCmd Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal cmd As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETSetCenterCmd Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal cmd As Short) As Integer
```

パラメータ

lineNo	***ライン番号*** (0始まり)
cmd	***コマンドデータ***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 参照)

注意

(4. 基本関数)の項を十分に理解頂いた上でご利用ください。

使用例

[VisualC++]

```
MNETSetCenterCmd(0, 0x1000);
```

[VisualBasic]

```
MNETSetCenterCmd 0, &H1000
```

[VB.NET]

```
MNETSetCenterCmd(0, &H1000)
```

ライン0のセンターデバイスにシステム通信コマンド(1000h)を書込みます。

5－1－5－2. MNETGetCenterSts

機能

センターデバイスステータス読出し

書式

[VisualC++]

```
short __stdcall MNETGetCenterSts(
      short lineNo,
      unsigned short* sts);
```

[VisualBasic]

```
Declare Function MNETGetCenterSts Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByRef sts As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETGetCenterSts Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByRef sts As Integer) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
sts	***取得データ格納変数***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 参照)

注意

(4. 基本関数)の項を十分に理解頂いた上でご利用ください。

使用例

[VisualC++]

```
unsigned short sts

MNETGetCenterSts(0, &sts);
```

[VisualBasic]

```
Dim sts As Integer

MNETGetCenterSts 0, sts
```

[VB.NET]

```
Dim sts As Short

MNETGetCenterSts(0, sts)
```

ライン0のセンターデバイスステータスを変数stsに取得します。

5－1－5－3. MNETGetCenterIntSts

機能

センターデバイス割込ステータス読出し

書式

[VisualC++]

```
short __stdcall MNETGetCenterIntSts(
      short lineNo,
      unsigned short* sts);
```

[VisualBasic]

```
Declare Function MNETGetCenterIntSts Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByRef sts As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETGetCenterIntSts Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByRef sts As Integer) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
sts	***取得データ格納変数***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 参照)

注意

(4. 基本関数)の項を十分に理解頂いた上でご利用ください。

使用例

[VisualC++]

```
unsigned short sts

MNETGetCenterIntSts(0, &sts);
```

[VisualBasic]

```
Dim sts As Integer

MNETGetCenterIntSts 0, sts
```

[VB.NET]

```
Dim sts As Short

MNETGetCenterIntSts(0, sts)
```

ライン0のセンターデバイス割込ステータスを変数stsに取得します。

5－1－5－4. MNETWriteCenterRegister

機能

センターデバイスレジスタ書込み

書式

[VisualC++]

```
short __stdcall MNETWriteCenterRegister(
      short lineNo,
      short cmd,
      unsigned short wdata);
```

[VisualBasic]

```
Declare Function MNETWriteCenterRegister Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal cmd As Integer,
      ByVal wdata As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETWriteCenterRegister Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal cmd As Short,
      ByVal wdata As Short) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
cmd	***書き込みコマンド*** センターデバイスLSI:G9001Aのレジスタ書込みコマンドを指定します。
wdata	***書込みデータ***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 参照)

注意

(4. 基本関数)の項を十分に理解頂いた上でご利用ください。

使用例

[VisualC++]

```
MNETWriteCenterRegister (0, MNET_CENTER_WRENV0, 0x100);
```

[VisualBasic]

```
MNETWriteCenterRegister 0, MNET_CENTER_WRENV0, &H100
```

[VB.NET]

```
MNETWriteCenterRegister(0, MNET_CENTER_WRENV0, &H100)
```

ライン0のセンターデバイスのRENV0レジスタに100hを書き込みます。

5－1－5－5. MNETReadCenterRegister

機能

センターデバイスレジスタ読出し

書式

[VisualC++]

```
short __stdcall MNETReadCenterRegister(
      short lineNo,
      short cmd,
      unsigned short* rdata);
```

[VisualBasic]

```
Declare Function MNETReadCenterRegister Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal cmd As Integer,
      ByRef rdata As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETReadCenterRegister Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal cmd As Short,
      ByRef rdata As Short) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

cmd ***読出しコマンド***

センターデバイスLSI:G9001Aのレジスタ読出しコマンドを指定します。

rdata ***読出しデータ***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 参照)

注意

(4. 基本関数)の項を十分に理解頂いた上でご利用ください。

使用例

[VisualC++]

```
unsigned short rdata

MNETReadCenterRegister (0, MNET_CENTER_RRERCNT, &rdata);
```

[VisualBasic]

```
Dim rdata As Integer

MNETReadCenterRegister 0, MNET_CENTER_RRERCNT ,rdata
```

[VB.NET]

```
Dim rdata As Short

MNETReadCenterRegister(0, MNET_CENTER_RRERCNT ,rdata)
```

ライン0のセンターデバイスのエラーカウンタを読み出します。

5－1－6. ローカルボード共通関数

5－1－6－1. MNETSetLocalData

機能

ローカルデバイスデータ書込み

書式

[VisualC++]

```
short __stdcall MNETSetLocalData(
      short lineNo,
      short devNo,
      short selectData,
      unsigned short wdata);
```

[VisualBasic]

```
Declare Function MNETSetLocalData Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal selectData As Integer,
      ByVal wdata As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETSetLocalData Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal selectData As Short,
      ByVal wdata As Short) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

selectData ***データ選択***

識別子	値	内容	設定可能範囲
MNET_LOCAL_PORT0	0	ポート0	0～FFh
MNET_LOCAL_PORT1	1	ポート1	0～FFh
MNET_LOCAL_PORT2	2	ポート2	0～FFh
MNET_LOCAL_PORT3	3	ポート3	0～FFh
MNET_LOCAL_ERRFLG	4	エラーフラグ	0/1
MNET_LOCAL_INTSET	5	入力変化割込設定	0～Fh
MNET_LOCAL_INTFLG	6	入力変化割込フラグ	0～Fh
MNET_LOCAL_DEVINFO	7	デバイス情報	0～FFh
MNET_LOCAL_PORTL	10	下位ポート(PORT1-0)	0～FFFFh
MNET_LOCAL_PORTH	11	上位ポート(PORT3-2)	0～FFFFh

wdata ***書込みデータ***

・上記の設定可能範囲を超える場合は設定値エラーとなります。

・未使用デバイスのポートデータ領域へのデータ設定はデフォルトではデバイス番号指定エラーとなります。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 参照)

Motionnet® Remote I/O & Remote Motion

使用例

[VisualC++]

```
MNETSetLocalData(0, 2, MNET_LOCAL_PORT2, 0xFF);
```

[VisualBasic]

```
MNETSetLocalData 0, 2, MNET_LOCAL_PORT2, &HFF
```

[VB.NET]

```
MNETSetLocalData(0, 2, MNET_LOCAL_PORT2, &HFF)
```

ライン0,デバイス2, PORT2にFFhを設定します。

5－1－6－2. MNETGetLocalData

機能

ローカルデバイスデータ読出し

書式

[VisualC++]

```
short __stdcall MNETGetLocalData(
        short lineNo,
        short devNo,
        short selectData,
        unsigned short* rdata);
```

[VisualBasic]

```
Declare Function MNETGetLocalData Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal selectData As Integer,
        ByRef rdata As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETGetLocalData Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal selectData As Short,
        ByRef rdata As Integer) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

selectData ***データ選択***

識別子	値	内容
MNET_LOCAL_PORT0	0	ポート0
MNET_LOCAL_PORT1	1	ポート1
MNET_LOCAL_PORT2	2	ポート2
MNET_LOCAL_PORT3	3	ポート3
MNET_LOCAL_ERRFLG	4	エラーフラグ
MNET_LOCAL_INTSET	5	入力変化割込設定
MNET_LOCAL_INTFLG	6	入力変化割込フラグ
MNET_LOCAL_DEVINFO	7	デバイス情報
MNET_LOCAL_PORTL	10	下位ポート(PORT1-0)
MNET_LOCAL_PORTH	11	上位ポート(PORT3-2)

rdata ***取得データ格納変数***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 参照)

Motionnet® Remote I/O & Remote Motion

使用例

[VisualC++]

```
unsigned short rdata

MNETGetLocalData(0, 2, MNET_LOCAL_PORT0, &rdata);
```

[VisualBasic]

```
Dim rdata as Integer

MNETGetLocalData 0, 2, MNET_LOCAL_PORT0, rdata
```

[VB.NET]

```
Dim rdata as Short

MNETGetLocalData(0, 2, MNET_LOCAL_PORT0, rdata)
```

ライン0,デバイス2,PORT0のデータを変数rdataに取得します。

5－1－6－3. MNETSetDLLPar

機能

DLL内部パラメータ設定

書式

[VisualC++]

```
short __stdcall MNETSetDLLPar(
        short lineNo,
        short devNo,
        short itemNo,
        char val);
```

[VisualBasic]

```
Declare Function MNETSetDLLPar Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal itemNo As Integer,
        ByVal val As Byte) As Integer
```

[VB.NET]

```
Declare Function MNETSetDLLPar Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal itemNo As Short,
        ByVal val As Byte) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
itemNo	***項目選択*** (5－3－2－1. DLL内部パラメータ項目番号定義) 参照
val	***設定値***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 参照)

備考

項目が「未使用デバイス出力ポートへのアクセスエラー判定」の場合はライン番号、デバイス番号の指定は無視されます。(何らかの値は設定ください)

使用例

[VisualC++]

```
MNETSetDLLPar(0, 0, MNET_DLLPAR_NU_ACCESS, 1);
```

[VisualBasic]

```
MNETSetDLLPar 0, 0, MNET_DLLPAR_NU_ACCESS, 1
```

[VB.NET]

```
MNETSetDLLPar(0, 0, MNET_DLLPAR_NU_ACCESS, 1)
```

DLL 内部パラメータ「未使用デバイス出力ポートへのアクセスエラー判定」を 1(エラーとしない)にセットします。

5−2. 機能操作説明

5−2−1. ローカルデバイスのデータ構成

Motionnet のローカルデバイスは機種を問わず、以下のデータを持っています。
各データの取得には MNETGetLocalData 関数、設定には MNETSetLocalData 関数を使用します。
対象データの選択は次の識別子を引数に指定します。

識別子	値	内容
MNET_LOCAL_PORT0	0	PORT0
MNET_LOCAL_PORT1	1	PORT1
MNET_LOCAL_PORT2	2	PORT2
MNET_LOCAL_PORT3	3	PORT3
MNET_LOCAL_ERRFLG	4	I/O通信エラーフラグ
MNET_LOCAL_INTSET	5	入力ポート変化割込設定
MNET_LOCAL_INTFLG	6	入力ポート変化割込フラグ
MNET_LOCAL_DEVINFO	7	デバイス情報

デバイス情報

デバイスの属性を示す 8 ビットのデータです。
取得されるデータは以下のようになります。

デバイス種類	データ
MNET-D340などの入力4ポートI/Oボード	84h
MNET-D322などの入力2ポート出力2ポートI/Oボード	82h
MNET-D304などの出力4ポート	80h
MNET-M101-DUMなどの1軸モーションポート	8Bh
MNET-BCDCシリーズなどの4軸モーションポート	8Bh
未接続	0

ポートの割付け

[I/O デバイスの場合]

各ポートが入力ポートか出力ポートであるかは、各ローカルデバイスのハード的な設定で決まっています。

PORT3	PORT2	PORT1	PORT0

[1 軸モーションデバイスの場合]

各ポートは以下のように割りつけられています。

PORT3 (汎用出力ポート)	PORT2 (汎用入力ポート)	PORT1 (ステータス上位)	PORT0 (ステータス下位)

5−2−2. ローカルデバイスのデータ取得

MNETGetLocalData関数を使用します。
3番目の引数で対象となるデータを指定します。

記述例

[VisualC++]

```
unsigned short rdata

MNETGetLocalData(0, 2, MNET_LOCAL_PORT0, &rdata);
```

[VisualBasic]

```
Dim rdata as Integer

MNETGetLocalData 0, 2, MNET_LOCAL_PORT0, rdata
```

[VB.NET]

```
Dim rdata as Short

MNETGetLocalData(0, 2, MNET_LOCAL_PORT0, rdata)
```

ライン0のデバイス2のPORT0のデータを変数rdataに取得します。

5−2−3. ローカルデバイスのデータ設定

MNETSetLocalData関数を使用します。
3番目の引数で対象となるデータを指定します。

記述例

[VisualC++]

```
MNETSetLocalData(0, 2, MNET_LOCAL_PORT2, 0xFF);
```

[VisualBasic]

```
MNETSetLocalData 0, 2, MNET_LOCAL_PORT2, &HFF
```

[VB.NET]

```
MNETSetLocalData(0, 2, MNET_LOCAL_PORT2, &HFF)
```

ライン0のデバイス2のPORT2にFFhを設定します。

5−2−4. I/O ポート変化割込

Motionnet には入力ポートの状態変化を示す「入力ポート変化割込フラグ」が用意されています。
この機能の利用により、入力ポートの状態監視を簡略化できます。

5−2−4−1. I/O ポート変化割込の設定

どのローカルデバイスの、どのポートの状態変化監視機能を有効にするかは各ローカルデバイスの「入力ポート変化割込設定」エリアに設定します。
設定を有効にしたいポートを 4 ビットで表現したデータ(0～15h)を設定します。
(出力ポートへの設定は無効となります)

PORT3	PORT2	PORT1 設定	PORT0 設定

上図のように PORT0,PORT1 の入力ポート変化割り込みを有効にしたい場合は設定値は 3h となります。
このデータの設定には MNETSetLocalData 関数を使用します。

記述例

[VisualC++]

```
MNETSetLocalData(0, 2, MNET_LOCAL_INTSET, 0x3);
```

[VisualBasic]

```
MNETSetLocalData 0, 2, MNET_LOCAL_INTSET, &H3
```

[VB.NET]

```
MNETSetLocalData(0, 2, MNET_LOCAL_INTSET, &H3)
```

ライン0のデバイス2の入力ポート変化割込設定に3hを設定します。

5−2−4−2. I/O ポート変化割込の確認とクリア

状態の確認

全 64 ローカルデバイス×4 ポート= 256 ポート分の入力ポート変化割込フラグの OR 状態はセンターデバイスのステータスの IOPC ビットに現れます。このビットの状態は MNETGetIOPC 関数で取得できます。

各ローカルデバイスの入力ポート変化割込フラグは「入力ポート変化割込設定」同様 4 ビットで構成されています。

PORT3	PORT2	PORT1 変化有	PORT0

PORT1 にポート状態の変化があった場合、上図のように PORT1 に対応するビットが ON します。
このデータの取得には MNETGetLocalData 関数を使用し、この場合取得されるデータは 2h となります。

記述例

[VisualC++]

```
short data;

MNETGetLocalData(0, 2, MNET_LOCAL_INTFLG, &data);
```

[VisualBasic]

```
Dim data As Integer

MNETGetLocalData 0, 2, MNET_LOCAL_INTFLG, data
```

[VB.NET]

```
Dim data As Short

MNETGetLocalData(0, 2, MNET_LOCAL_INTFLG, data)
```

ライン0のデバイス2の入力ポート変化割込フラグを変数dataに取得します。

フラグのクリア

フラグが ON したままであると次に入力ポートの状態に変化があっても、フラグ状態からは変化の確認を行えません。したがってフラグをクリアする必要があります。
フラグのクリアは、MNETSetLocalData 関数を使用し、ON であるビットに'1'を書込むことで行われます。

記述例

[VisualC++]

```
MNETSetLocalData(0, 2, MNET_LOCAL_INTFLG, 0x2);
```

[VisualBasic]

```
MNETSetLocalData 0, 2, MNET_LOCAL_INTFLG, &h2
```

[VB.NET]

```
MNETSetLocalData(0, 2, MNET_LOCAL_INTFLG, &h2)
```

ライン0, デバイス2, PORT1の入力ポート変化割込フラグをクリアします。

5−2−5. 通信エラー

Motionnet に発生する通信エラー状況を取得するためのデータとして次の 2 つが用意されております。

(1) I/O 通信エラーフラグ

I/O 通信時に発生する通信エラーフラグです。

各ローカルデバイスは I/O 通信エラーフラグを 1 ビット持っており、I/O 通信エラーが 3 サイクル連続した場合に ON します。

(2) センターデバイス エラーカウンタ

センターデバイスに装備されているエラーカウンタです。

エラーカウンタには、I/O 通信エラーフラグに現れない連続回数が 3 回以下の I/O 通信エラー、データ通信エラーなど Motionnet の全ての通信エラーがハード的にカウントされます。

5−2−5−1. I/O 通信エラーフラグの確認とクリア

状態の確認

各ローカルデバイスの I/O 通信エラーフラグの取得には MNETGetLocalData 関数を使用します。

またライン上全 64 ローカルデバイスの I/O 通信エラーフラグの OR 状態はセンターデバイスのステータス EIOE ビットに現れます。このビットの状態は MNETGetEIOE 関数で取得できます。

記述例

[VisualC++]

```
short data;

MNETGetLocalData(0, 2, MNET_LOCAL_ERRFLG, &data);
```

[VisualBasic]

```
Dim data As Integer

MNETGetLocalData 0, 2, MNET_LOCAL_ERRFLG, data
```

[VB.NET]

```
Dim data As Short

MNETGetLocalData(0, 2, MNET_LOCAL_ERRFLG, data)
```

フラグのクリア

フラグが ON したままであると、たとえエラー状態が解除されていたとしてもフラグはクリアされず、次にエラー状態が発生しても、フラグ状態からはエラーの発生の確認を行えません。したがってフラグをクリアする必要があります。

フラグのクリアは、MNETSetLocalData 関数を使用し、ON であるビットに'1'を書込むことで行われます。

記述例

[VisualC++]

```
MNETSetLocalData(0, 2, MNET_LOCAL_ERRFLG, 1);
```

[VisualBasic]

```
MNETSetLocalData 0, 2, MNET_LOCAL_ ERRFLG, 1
```

[VB.NET]

```
MNETSetLocalData(0, 2, MNET_LOCAL_ ERRFLG, 1)
```

ライン0, デバイス2のI/O通信エラーフラグをクリアします。

5-2-5-2. センターデバイス エラーカウンタの取得とクリア

カウンタの取得

センターデバイス エラーカウンタの取得は MNETReadCenterRegister 関数(注)で行ないます。

このエラーカウンタはシステム通信の無応答などもカウントするため、例えばローカルデバイスをライン上 2 個接続してシステム通信を行なった場合は、62 個(64 − 2 個) × 4 回(1＋リトライ 3 回) = 248 回の通信エラーがカウントされますが、再度カウンタクリアをすることで 0 からカウントされます。

またカウンタの最大値は 65535 となっております。

記述例

[VisualC++]

```
unsigned short errCnt;

MNETReadCenterRegister(0, MNET_CENTER_RRERCNT ,&errCnt);
```

[VisualBasic]

```
Dim errCnt As Integer

MNETReadCenterRegister 0, MNET_CENTER_RRERCNT ,errCnt
```

[VB.NET]

```
Dim errCnt As Short

MNETReadCenterRegister(0, MNET_CENTER_RRERCNT ,errCnt)
```

ライン0のセンターデバイス エラーカウンタ値を取得します。

カウンタのクリア

センターデバイス エラーカウンタのクリアは MNETSetCenterCmd 関数(注)で行ないます。

記述例

[VisualC++]

```
MNETSetCenterCmd(0, MNET_CENTER_CLR_ERCNT);
```

[VisualBasic]

```
MNETSetCenterCmd 0, MNET_CENTER_CLR_ERCNT
```

[VB.NET]

```
MNETSetCenterCmd(0, MNET_CENTER_CLR_ERCNT)
```

ライン0のセンターデバイス エラーカウンタをクリアします。

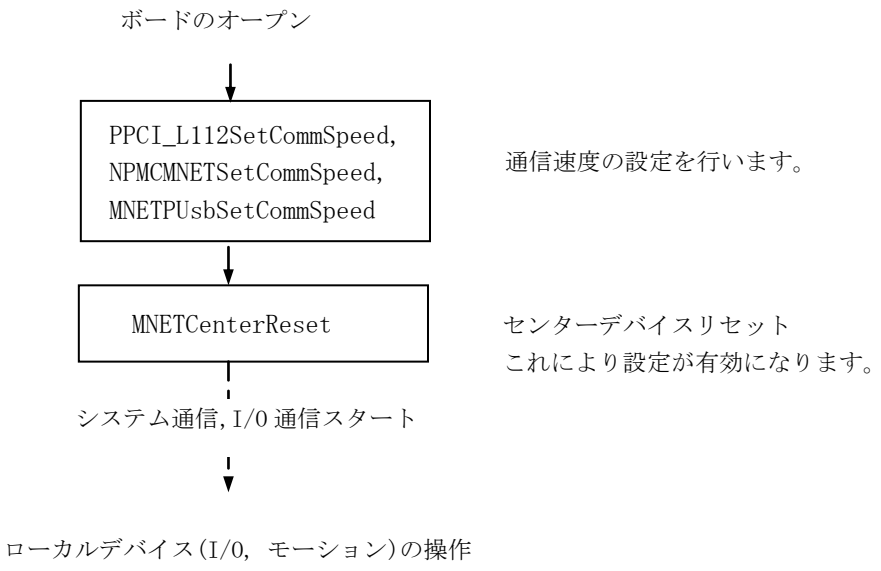
(注)これらの関数はセンターデバイス基本関数として区分され、4.基本関数の項で複合関数との混在を避けるように注意しておりますが、この用法におきましては混在されても問題ありません。

5−2−6. 通信速度の設定

通信速度の設定は PPCI_L112SetCommSpeed 関数, NPMCMNETSetCommSpeed 関数, MNETPUsbSetCommSpeed 関数で行います。設定はかならず Motionnet 通信開始前に行ってください。

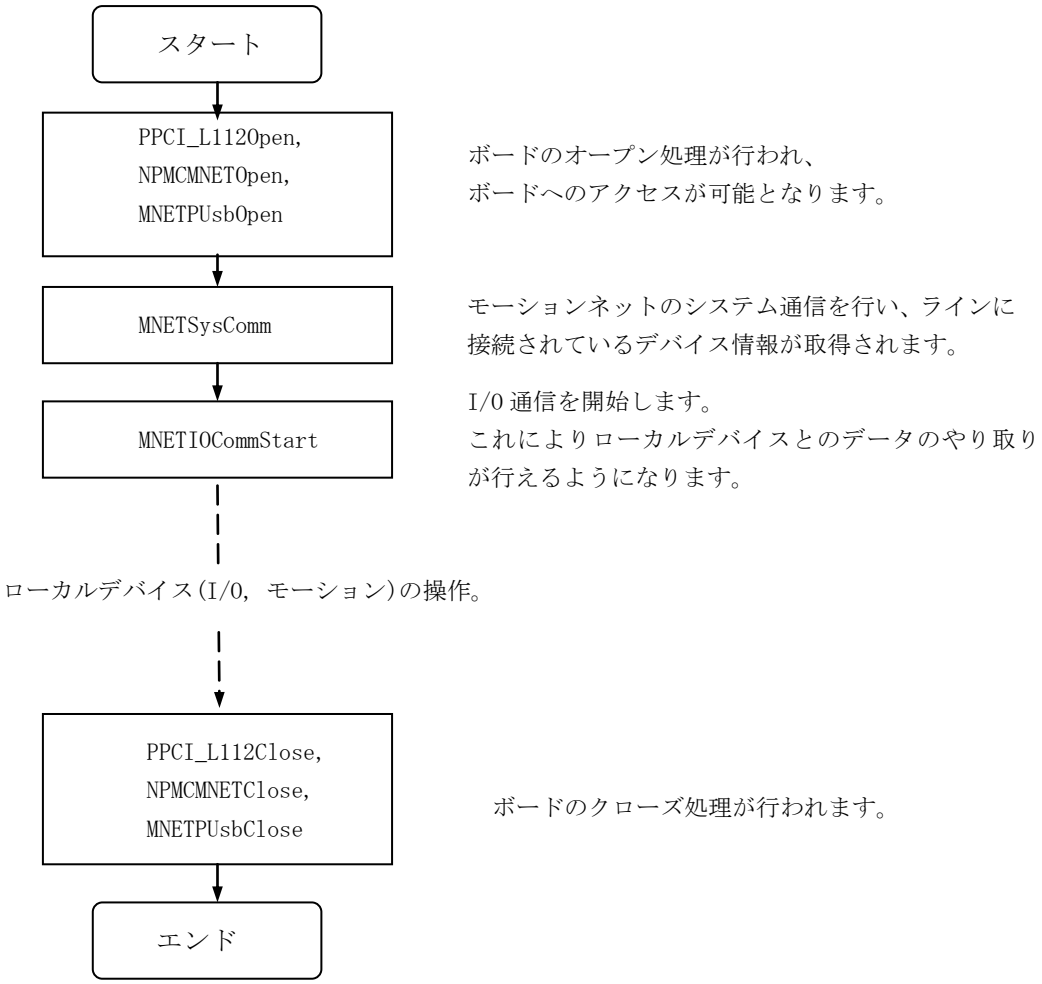
PPCI_L112SetCommSpeed 関数, NPMCMNETSetCommSpeed 関数, MNETPUsbSetCommSpeed 関数実行後に、MNETCenterReset 関数でセンターデバイスのリセットを行ってください。　これにより設定が有効になります。　手順を以下に示します。

なお、ボードオープン後この関数を実行しない場合、設定初期値は 20Mbps となります。



5−2−7. 全体の流れ

PPCI_L112.DLL, NPMCMNET104.DLL, MNETPUSB.DLL 操作の流れは次のようになります。



5−2−7−1. I/O デバイスプログラム概略

Motionnet の I/O デバイスを操作するプログラムコードの概略を示します。
以下のコードはあくまで一連の流れをつかんでもらうためのものであり、関数の実行結果の取得などをはじめとする実際の動作において必要な処理の多くを省略してありますのでご注意ください。
またボードのオープン/クローズ、リセット、システム通信、I/O 通信スタートなどの処理は通常、システム立ち上がり時に 1 回だけ行う処理であり、操作のたびに毎回行う処理ではありません。

[VisualC++]

```
short lineNo;           //ライン番号
short devNo;            //デバイス番号
short portNo;           //ポート番号(項目番号)
unsigned short data[4]; //取得データ格納

PPCI_L112Open();                          // ボードのオープン

lineNo = 0;
MNETCenterReset(lineNo);                  // センターデバイスリセット
MNETSysComm(lineNo);                      // システム通信
MNETIOCommStart(lineNo);                  // I/O 通信スタート

for (portNo = 0; portNo <= 3; portNo++) {
MNETGetLocalData(lineNo, devNo, portNo, &data[portNo]); // ポートデータ読出し
}

devNo = 0;
portNo = 2;
MNETSetLocalData(lineNo, devNo, portNo, 0xff);            // ポートデータ書込み

MNETCenterReset(lineNo);                  // センターデバイスリセット
PPCI_L112Close();                         // ボードのクローズ
```

[VisualBasic]

```
Dim lineNo As Integer   ’ライン番号
Dim devNo As Integer    ’デバイス番号
Dim portNo As Integer   ’ポート番号(項目番号)
Dim data(0 To 3)        ’取得データ格納

PPCI_L112Open           ’ボードのオープン

lineNo = 0
MNETCenterReset lineNo              ’センターデバイスリセット
MNETSysComm lineNo                  ’システム通信
MNETIOCommStart lineNo              ’I/O 通信スタート

For portNo = 0 To 3
MNETGetLocalData lineNo, devNo, portNo, data(portNo) ’ポートデータ読出し
Next portNo

devNo = 0
portNo = 2
MNETSetLocalData  lineNo, devNo, portNo, &HFF          ’ポートデータ書込み

MNETCenterReset 0                   ’センターデバイスリセット
PPCI_L112Close                      ’ボードのクローズ
```

[VB.NET]

```
Dim lineNo As Short   ’ライン番号
Dim devNo As Short    ’デバイス番号
Dim portNo As Short   ’ポート番号(項目番号)
Dim data(0 To 3)      ’取得データ格納

PPCI_L112Open()           ’ボードのオープン

lineNo = 0
MNETCenterReset(lineNo)             ’センターデバイスリセット
MNETSysComm(lineNo)                 ’システム通信
MNETIOCommStart(lineNo)             ’I/O 通信スタート

For portNo = 0 To 3
MNETGetLocalData(lineNo, devNo, portNo, data(portNo)) ’ポートデータ読出し
Next portNo

devNo = 0
portNo = 2
MNETSetLocalData(lineNo, devNo, portNo, &HFF)          ’ポートデータ書込み

MNETCenterReset(0)                  ’センターデバイスリセット
PPCI_L112Close()                    ’ボードのクローズ
```

5−3. データ定義

5−3−1. ステータス／ビットマップ

5−3−1−1. センターデバイスステータス

MNETGetCenterSts 関数で取得できるセンターデバイスのステータスのマップです。

Bit	ﾋﾞｯﾄ名	内容
0	CEND	データ送信用FIFO書き込み可能時に1になります。 システム通信、またはデータ通信が完了して、データ送信用FIFOに次データの書き込みが可能になった時に1なり、割り込み信号(INT)を出力します。 本ステータス読み出し後に0に戻ります。
1	BRKF	ブレークフレーム受信時に1になり、割り込み信号(INT)を出力します。 本ステータス読み出し後に0に戻ります。
2	IOPC	「入力変化割込設定」を1にセットした入力ポートの状態が変化した時に1になり、割り込み信号(INT)を出力します。 「入力ポート変化割込フラグ」の全256ビットのOR信号です。 全ビットが0になると、このビットは0に戻ります。
3	EIOE	I/O通信エラー発生時に1になり、割り込み信号(INT)を出力します。 「I/O通信エラーフラグ」の全64ビットのOR信号です。 全ビットが0になると、このビットは0に戻ります。
4	EDTE	データ通信、またはシステム通信エラー発生時に1になり、割り込み信号(INT)を出力します。 本ステータス読み出し後に0に戻ります。
5	ERAE	ローカル側受信処理エラー発生時に1になり、割り込み信号(INT)を出力します。 本ステータス読み出し後に0に戻ります。 エラー発生デバイスNo.と受信処理エラーの内容は割り込みステータスで確認します。注1
6	CAER	CPUのアクセスエラーです。 送信データが空のままでデータ送信コマンドを書き込む等、CPUから不適切なアクセスがあると1になり、割り込み信号(INT)を出力します。 本ステータス読み出し後に0に戻ります。 エラー発生の内容は割り込みステータスで確認します。
7	未定義	常に0になります。
8	REF	未送信の出力ポートデータが有る時1になります。 出力ポートエリアに書き込むと1になり、全ポートへのサイクリック通信が終了すると0に戻ります。
9	TDRB	データ送信用FIFOに送信データがある時に1になります。 データ送信用FIFOに書き込むと1になり、データ送信コマンド、または送信用FIFOリセットコマンドを書き込んだ時に0に戻ります。
10	RDBB	データ受信用FIFOに受信データがある時に1になります。 データデバイスからデータを受信すると1になり、CPUが受信データを全て読み出すと0に戻ります。
11	未定義	常に0になります。
12	SBSY	I/O通信（サイクリック通信）スタート中に1になります。
13	RBSY	リセット処理中に1になります。
14	DBSY	システム通信中、またはデータ通信中に1になります。
15	未定義	常に0になります。

注1：I/Oデバイスに対して、PMD2～0の設定(ポート入出力設定)と違う形式の通信を行うことで発生するエラーは、割込ステータスで、デバイス番号の確認ができます（複数デバイスにエラーが発生した場合は、最後にエラーになったデバイス番号になります）。

5−3−1−2. センターデバイス割り込みステータス

MNETGetCenterIntSts 関数で取得できるセンターデバイスの割り込みステータスのマップです。

Bit	ﾋﾞｯﾄ名	内容
5～0	EDN5～0	ステータスの、EDTE=1または、ERAE=1（ERA=0001：PMD設定(ポート入出力設定)と違うI/Oデータ受信によるエラー時）のデバイス番号で、次回のエラー発生まで記憶されます。
6	未定義	常に0になります。
7	LNRV	ローカル側データ未受信時に1になります。 データ通信、またはシステム通信（属性情報の取得時のみ）がエラーで終了した(EDTE=1)場合、ローカル側がセンターからのデータを受信出来なかった時（ローカル側は無応答）は1になり、受信出来た時は0になります。 次回のエラー発生まで記憶されます。
11～8	ERA3～0	ローカルデバイス側のデータ受信処理エラーのコードで、次回のエラー発生まで記憶されます。 0001: PMD設定(ポート入出力設定)と違うI/Oデータ受信時 0010: I/Oデバイスがデータ通信フレーム受信時（＊） 0011: データデバイスが受信バッファ容量以上のフレーム受信時（＊）
15～12	CAE3～0	CPUからのアクセスエラーのコードで、次回のエラー発生まで記憶されます。 0001: 使用デバイス0でI/O通信スタートコマンド書き込み時 0010: 送信データ無しでデータ送信スタートコマンド書き込み時 0011: DBSY=1の時に、①②のアクセスを行った時 ①送受信用FIFOの読み出し、または書き込み ②システム通信、またはデータ通信スタートコマンドを書き込んだ時 0100: 未使用デバイスに対してデータを送信した時

（＊）ERA コードが“0010”、“0011”の場合、そのデバイス番号は EDN へ反映されません。

Motionnet®
Remote I/O & Remote Motion

5−3−2. 識別子

5−3−2−1. DLL 内部パラメータ項目番号定義

MNETSetDLLPar関数使用時に対象のパラメータ項目を指定します。

識別子	値	内容	設定範囲	設定値／取得値（網掛け：デフォルト）
MNET_DLLPAR_NU_ACCESS	2E0h	未使用デバイスのポートデータの書込み(注)	0/1	0:エラー,1:エラーとしない

（注）未使用デバイスのポートデータの読込み、およびその他 I/O 通信エラーフラグ、入出力変化割込設定/フラグ、デバイス情報のデータは読込み/書込みはこのパラメータ設定に関わらず有効です。(エラーとなりません)

5−3−2−2. センターデバイスコマンド

MNETSetCenterCmd 関数を使用し、センターデバイスの制御などを行う際に指定する識別子です。
(4. 基本関数)の項を十分に理解頂いた上でご利用ください。

識別子	値	内容
MNET_CENTER_RESET	100h	リセット
MNET_CENTER_SYSCOM	1000h	システム通信
MNET_CENTER_STA_IOCOM	3000h	I/O通信スタート
MNET_CENTER_STP_IOCOM	3100h	I/O通信ストップ
MNET_CENTER_CLR_STS	400h	ステータスクリア
MNET_CENTER_CLR_ERCNT	600h	エラーカウンタクリア
MNET_CENTER_BRKCOM	610h	ブレーク通信発生

5−3−2−3. センターデバイスレジスタコマンド

MNETWriteCenterRegister 関数、MNETReadCenterRegister 関数を使用し、センターデバイスのレジスタ読書きを行う際に指定する識別子です。
(4. 基本関数)の項を十分に理解頂いた上でご利用ください。

識別子	値	内容
MNET_CENTER_WRENVO	5500h	RENVO書き込み
MNET_CENTER_RRENVO	6500h	RENVO読み出し
MNET_CENTER_RRERCNT	6501h	エラーカウンタ読み出し
MNET_CENTER_RRCYCLE	6502h	サイクリック周期読み出し
MNET_CENTER_RRDADR	6503h	最終データ通信デバイスアドレス読み出し
MNET_CENTER_RRVER	6504h	バージョン読み出し

5−3−3. 戻り値

識別子	値	内容
MNET_SUCCESS	0	エラー無し
MNET_ERR_SET_CARDNO	0100h	ボード番号指定エラー
MNET_ERR_SET_LINENO	0101h	ライン番号指定エラー
MNET_ERR_SET_DEVNO	0102h	デバイス番号指定エラー
MNET_ERR_SET_PORTNO	0103h	ポート番号指定エラー
MNET_ERR_SET_ITEM	0104h	項目指定エラー
MNET_ERR_SET_VAL	0105h	設定値エラー
MNET_ERR_DEV_NOT_USED	0106h	未使用デバイスアクセスエラー
MNET_ERR_NO_CARD	1000h	ボード未接続。ボードが認識されません
MNET_ERR_IOCOMM_NOT_START	1001h	I/O通信未スタート
MNET_ERR_DATACOMM_FAILED	1002h	データ通信エラー(EDTE)
MNET_ERR_DATACOMM_PROC	1003h	データ通信処理エラー(ERAE,CAER)
MNET_ERR_CENTER_TIMEOUT	1004h	I/O通信開始タイムアウト
MNET_ERR_PUSB_DISCONNECT	10h	USB切断

6. ローカル１軸モーションボード機能

6－1. 関数詳細

6－1－1. ローカル１軸モーションボード関数

6－1－1－1. MNETM1SetSpeed

機能

速度データ設定

書式

[VisualC++]

```
short __stdcall MNETM1SetSpeed(
      short lineNo,
      short devNo,
      TMNETM1Cmd* cmd);
```

[VisualBasic]

```
Declare Function MNETM1SetSpeed Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByRef cmd As TMNETM1Cmd) As Integer
```

[VB.NET]

```
Declare Function MNETM1SetSpeed Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByRef cmd As TMNETM1Cmd) As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
cmd	***速度データ構造体*** （6－2－3.　速度データ構造体　参照）

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照）

使用例

例① スタートモード : FL定速

[VisualC++]

```
TMNETM1Cmd cmd = {0};              // 速度データ構造体定義, 初期化(0リセット)

cmd.StartMode = MNET_STAMODE_FL;  // スタートモード設定
cmd.SpdL = 100;                    // FL速度   :   100PPS
MNETM1SetSpeed(0, 3, &cmd);
```

[VisualBasic]

```
Dim cmd As TMNETM1Cmd              ’速度データ構造体定義, 初期化(0リセット)

cmd.StartMode = MNET_STAMODE_FL       ’スタートモード設定
cmd.SpdL = 100                        ’FL速度   :   100PPS
MNETM1SetSpeed 0, 3, cmd
```

[VB.NET]

```
Dim cmd As TMNETM1Cmd              ’速度データ構造体定義, 初期化(0リセット)

cmd.StartMode = MNET_STAMODE_FL       ’スタートモード設定
cmd.SpdL = 100                        ’FL速度   :   100PPS
MNETM1SetSpeed(0, 3, cmd)
```

ライン0, デバイス3の速度データを スタートモード:FL定速, FL速度:100PPSに設定します。

例② スタートモード ： 加減速, 加減速指定方法 ： 時間

[VisualC++]

```
TMNETM1Cmd cmd = {0};              // 速度データ構造体定義, 初期化(0リセット)

cmd.StartMode = MNET_STAMODE_UD;  // スタートモード設定
cmd.SpdL = 100;                   // FL速度    : 100PPS
cmd.SpdH = 3000;                  // FH速度    : 3000PPS
cmd.AdcMtd = MNET_ADC_BY_TIME;    // 加減速指定方法設定
cmd.Acc = 0.3;                    // 加速時間 : 0.3秒
cmd.Dec = 0.3;                    // 減速時間 : 0.3秒
MNETM1SetSpeed(0, 3, &cmd);
```

[VisualBasic]

```
Dim cmd As TMNETM1Cmd             ’速度データ構造体定義, 初期化(0リセット)

cmd.StartMode = MNET_STAMODE_UD   ’スタートモード設定
cmd.SpdL = 100                    ’FL速度    : 100PPS
cmd.SpdH = 3000                   ’FH速度    : 3000PPS
cmd.AdcMtd = MNET_ADC_BY_TIME     ’加減速指定方法設定
cmd.Acc = 0.3                     ’加速時間 : 0.3秒
cmd.Dec = 0.3                     ’減速時間 : 0.3秒
MNETM1SetSpeed 0, 3, cmd
```

[VB.NET]

```
Dim cmd As TMNETM1Cmd             ’速度データ構造体定義, 初期化(0リセット)

cmd.StartMode = MNET_STAMODE_UD   ’スタートモード設定
cmd.SpdL = 100                    ’FL速度    : 100PPS
cmd.SpdH = 3000                   ’FH速度    : 3000PPS
cmd.AdcMtd = MNET_ADC_BY_TIME     ’加減速指定方法設定
cmd.Acc = 0.3                     ’加速時間 : 0.3秒
cmd.Dec = 0.3                     ’減速時間 : 0.3秒
MNETM1SetSpeed(0, 3, cmd)
```

ライン0, デバイス3の速度データを スタートモード:加減速, FL速度:100PPS, FH速度:3000PPS, 加速時間:0.3秒, 減速時間:0.3秒に設定します。

例③ スタートモード ： 加減速，加減速指定方法 ： レート、S字動作

[VisualC++]

```
TMNETM1Cmd cmd = {0};              // 速度データ構造体定義，初期化(0リセット)

cmd.StartMode = MNET_STAMODE_UD;  // スタートモード設定
cmd.SpdL = 100;                   // FL速度    ： 100PPS
cmd.SpdH = 3000;                  // FH速度    ： 3000PPS
cmd.AdcMtd = MNET_ADC_BY_RATE;    // 加減速指定方法設定
cmd.Acc = 1000;                   // 加速レート ： 1000パルス/S
cmd.Dec = 1000;                   // 減速レート ： 1000パルス/S
cmd.Smode = MNETM1_USE_S;         // S字使用
MNETM1SetSpeed(0, 3, &cmd);
```

[VisualBasic]

```
Dim cmd As TMNETM1Cmd              ’速度データ構造体定義，初期化(0リセット)

cmd.StartMode = MNET_STAMODE_UD    ’スタートモード設定
cmd.SpdL = 100                     ’FL速度    ： 100PPS
cmd.SpdH = 300                     ’FH速度    ： 3000PPS
cmd.AdcMtd = MNET_ADC_BY_RATE      ’加減速指定方法設定
cmd.Acc = 1000                     ’加速レート ： 1000パルス/S
cmd.Dec = 1000                     ’減速レート ： 1000パルス/S
cmd.Smode = MNETM1_USE_S           ’S字使用
MNETM1SetSpeed 0, 3, cmd
```

[VB.NET]

```
Dim cmd As TMNETM1Cmd              ’速度データ構造体定義，初期化(0リセット)

cmd.StartMode = MNET_STAMODE_UD    ’スタートモード設定
cmd.SpdL = 100                     ’FL速度    ： 100PPS
cmd.SpdH = 300                     ’FH速度    ： 3000PPS
cmd.AdcMtd = MNET_ADC_BY_RATE      ’加減速指定方法設定
cmd.Acc = 1000                     ’加速レート ： 1000パルス/S
cmd.Dec = 1000                     ’減速レート ： 1000パルス/S
cmd.Smode = MNETM1_USE_S           ’S字使用
MNETM1SetSpeed(0, 3, cmd)
```

ライン0，デバイス3の速度データを スタートモード:加減速，FL速度:100PPS，FH速度:3000PPS，加速レート:1000PPS/S，減速レート:1000PPS/S，S字動作有りに設定します。

6－1－1－2. MNETM1Inc

機能

Inc移動(相対位置移動)を起動します。
(スタートコマンドを発行します)

書式

[VisualC++]

```
short __stdcall MNETM1Inc(
        short lineNo,
        short devNo,
        short startMode,
        long amount);
```

[VisualBasic]

```
Declare Function MNETM1Inc Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal startMode As Integer,
        ByVal amount As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM1Inc Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal startMode As Short,
        ByVal amount As Long) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

startMode ***スタートモード***

識別子	値	内容
MNET_STAMODE_FL	0050h	FL定速
MNET_STAMODE_FH	0051h	FH 定速
MNET_STAMODE_UD	0053h	加減速

Amount ***移動量***

備考

速度データはMNETM1SetSpeed関数で設定ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

例① スタートモード ： FL定速

[VisualC++]

```
MNETM1Inc(0, 3, MNET_STAMODE_FL, 5000);
```

[VisualBasic]

```
MNETM1Inc 0, 3, MNET_STAMODE_FL, 5000
```

[VB.NET]

```
MNETM1Inc(0, 3, MNET_STAMODE_FL, 5000)
```

ライン0，デバイス3に移動量5000パルス，スタートモード:FL定速でInc移動指令を発行します。

例② スタートモード ： 加減速

[VisualC++]

```
MNETM1Inc(0, 3, MNET_STAMODE_UD, -300);
```

[VisualBasic]

```
MNETM1Inc 0, 3, MNET_STAMODE_UD, -300
```

[VB.NET]

```
MNETM1Inc(0, 3, MNET_STAMODE_UD, -300)
```

ライン0，デバイス3に移動量-300パルス，スタートモード:加減速でInc移動指令を発行します。

6－1－1－3. MNETM1Abs

機能

Abs移動(絶対位置移動)(カウンタ1基準)を起動します。
(スタートコマンドを発行します)

書式

[VisualC++]

```
short __stdcall MNETM1Abs(
        short lineNo,
        short devNo,
        short startMode,
        long pos);
```

[VisualBasic]

```
Declare Function MNETM1Abs Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal startMode As Integer,
        ByVal pos As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM1Abs Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal startMode As Short,
        ByVal pos As Long) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号***

startMode ***スタートモード***

識別子	値	内容
MNET_STAMODE_FL	0050h	FL定速
MNET_STAMODE_FH	0051h	FH 定速
MNET_STAMODE_UD	0053h	加減速

pos ***目標位置***

備考

速度データはMNETM1SetSpeed関数で設定ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

例① スタートモード ： FL定速

[VisualC++]

```
MNETM1Abs(0, 3, MNET_STAMODE_FL, 5000);
```

[VisualBasic]

```
MNETM1Abs 0, 3, MNET_STAMODE_FL, 5000
```

[VB.NET]

```
MNETM1Abs(0, 3, MNET_STAMODE_FL, 5000)
```

ライン0, デバイス3に目標値5000(カウンタ1基準), スタートモード:FL定速で
Abs移動指令を発行します。

例② スタートモード ： 加減速

[VisualC++]

```
MNETM1Abs(0, 3, MNET_STAMODE_UD, -300);
```

[VisualBasic]

```
MNETM1Abs 0, 3, MNET_STAMODE_UD, -300
```

[VB.NET]

```
MNETM1Abs(0, 3, MNET_STAMODE_UD, -300)
```

ライン0, デバイス3に目標値-300(カウンタ1基準), スタートモード:加減速で
Abs移動指令を発行します。

6－1－1－4. MNETM1Continue

機能

連続移動を起動します。
(スタートコマンドを発行します)

書式

[VisualC++]

```
short __stdcall MNETM1Continue(
        short lineNo,
        short devNo,
        short startMode,
        short dir);
```

[VisualBasic]

```
Declare Function MNETM1Continue Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal startMode As Integer,
        ByVal dir As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM1Continue Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal startMode As Short,
        ByVal dir As Integer) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

startMode ***スタートモード***

識別子	値	内容
MNET_STAMODE_FL	0050h	FL定速
MNET_STAMODE_FH	0051h	FH 定速
MNET_STAMODE_UD	0053h	加減速

dir ***動作方向***

識別子	値	内容
MNET_DIR_PLUS	0	プラス方向
MNET_DIR_MINUS	1	マイナス方向

備考

速度データはMNETM1SetSpeed関数で設定ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

例① スタートモード ： FL定速

[VisualC++]

```
MNETM1Continue(0, 3, MNET_STAMODE_FL, MNET_DIR_PLUS);
```

[VisualBasic]

```
MNETM1Continue 0, 3, MNET_STAMODE_FL, MNET_DIR_PLUS
```

[VB.NET]

```
MNETM1Continue(0, 3, MNET_STAMODE_FL, MNET_DIR_PLUS)
```

ライン0, デバイス3に動作方向:プラス, スタートモード:FL定速で
連続移動指令を発行します。

例② スタートモード ： 加減速

[VisualC++]

```
MNETM1Continue (0, 3, MNET_STAMODE_UD, MNET_DIR_MINUS);
```

[VisualBasic]

```
MNETM1Continue 0, 3, MNET_STAMODE_UD, MNET_DIR_MINUS
```

[VB.NET]

```
MNETM1Continue(0, 3, MNET_STAMODE_UD, MNET_DIR_MINUS)
```

ライン0, デバイス3に動作方向:マイナス, スタートモード:加減速で
連続移動指令を発行します。

6－1－1－5. MNETM1Org

機能

ORG移動(原点復帰移動)を起動します。
(スタートコマンドを発行します)

書式

[VisualC++]

```
short __stdcall MNETM1Org(
        short lineNo,
        short devNo,
        short startMode,
        short dir);
```

[VisualBasic]

```
Declare Function MNETM1Org Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal startMode As Integer,
        ByVal dir As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM1Org Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal startMode As Short,
        ByVal dir As Integer) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

startMode ***スタートモード***

識別子	値	内容
MNET_STAMODE_FL	0050h	FL定速
MNET_STAMODE_FH	0051h	FH 定速
MNET_STAMODE_UD	0053h	加減速

dir ***動作方向***

識別子	値	内容
MNET_DIR_PLUS	0	プラス方向
MNET_DIR_MINUS	1	マイナス方向

備考

速度データはMNETM1SetSpeed関数で設定ください。
原点復帰モードなどはMNETM1SetPar関数で設定します。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

例　スタートモード　：　FL定速

[VisualC++]

```
MNETM1Org(0, 3, MNET_STAMODE_FL, MNET_DIR_MINUS);
```

[VisualBasic]

```
MNETM1Org 0, 3, MNET_STAMODE_FL, MNET_DIR_MINUS
```

[VB.NET]

```
MNETM1Org(0, 3, MNET_STAMODE_FL, MNET_DIR_MINUS)
```

ライン0, デバイス3に動作方向: マイナス, スタートモード:FL定速で
ORG移動(原点復帰移動)指令を発行します。

6－1－1－6. MNETM1EscOrg

機能

原点脱出移動を起動します。
(スタートコマンドを発行します)

書式

[VisualC++]

```
short __stdcall MNETM1EscOrg(
        short lineNo,
        short devNo,
        short startMode,
        short dir);
```

[VisualBasic]

```
Declare Function MNETM1EscOrg Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal startMode As Integer,
        ByVal dir As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM1EscOrg Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal startMode As Short,
        ByVal dir As Integer) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

startMode ***スタートモード***

識別子	値	内容
MNET_STAMODE_FL	0050h	FL定速
MNET_STAMODE_FH	0051h	FH 定速

加減速動作は設定できません。

Dir ***動作方向***

識別子	値	内容
MNET_DIR_PLUS	0	プラス方向
MNET_DIR_MINUS	1	マイナス方向

備考

速度データはMNETM1SetSpeed関数で設定ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

例　スタートモード　：　FL定速

[VisualC++]

```
MNETM1EscOrg(0, 3, MNET_STAMODE_FL, MNET_ DIR_PLUS);
```

[VisualBasic]

```
MNETM1EscOrg 0, 3, MNET_STAMODE_FL, MNET_ DIR_PLUS
```

[VB.NET]

```
MNETM1EscOrg(0, 3, MNET_STAMODE_FL, MNET_ DIR_PLUS)
```

ライン0, デバイス3に動作方向: プラス, スタートモード:FL定速で
原点脱出移動指令を発行します。

6－1－1－7. MNETM1ToEL

機能

EL(エンドリミット)までの移動を起動します。
(スタートコマンドを発行します)

書式

[VisualC++]

```
short __stdcall MNETM1ToEL(
        short lineNo,
        short devNo,
        short startMode,
        short dir);
```

[VisualBasic]

```
Declare Function MNETM1ToEL Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal startMode As Integer,
        ByVal dir As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM1ToEL Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal startMode As Short,
        ByVal dir As Integer) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

startMode ***スタートモード***

識別子	値	内容
MNET_STAMODE_FL	0050h	FL定速
MNET_STAMODE_FH	0051h	FH 定速
MNET_STAMODE_UD	0053h	加減速

dir ***動作方向***

識別子	値	内容
MNET_DIR_PLUS	0	プラス方向
MNET_DIR_MINUS	1	マイナス方向

備考

速度データはMNETM1SetSpeed関数で設定ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

例　スタートモード　：　FL定速

[VisualC++]

```
MNETM1ToEL(0, 3, MNET_STAMODE_FL, MNET_ DIR_PLUS);
```

[VisualBasic]

```
MNETM1ToEL 0, 3, MNET_STAMODE_FL, MNET_ DIR_PLUS
```

[VB.NET]

```
MNETM1ToEL(0, 3, MNET_STAMODE_FL, MNET_ DIR_PLUS)
```

ライン0, デバイス3に動作方向: プラス, スタートモード:FL定速でEL(エンドリミット)までの移動指令を発行します。
(プラスエンドリミットまたはプラスソフトリミットまでの移動となります)

6－1－1－8. MNETM1EscEL

機能

EL(エンドリミット)からの脱出移動を起動します。
(スタートコマンドを発行します)

書式

[VisualC++]

```
short __stdcall MNETM1EscEL(
        short lineNo,
        short devNo,
        short startMode,
        short dir);
```

[VisualBasic]

```
Declare Function MNETM1EscEL Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal startMode As Integer,
        ByVal dir As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM1EscEL Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal startMode As Short,
        ByVal dir As Integer) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

startMode ***スタートモード***

識別子	値	内容
MNET_STAMODE_FL	0050h	FL定速
MNET_STAMODE_FH	0051h	FH 定速
MNET_STAMODE_UD	0053h	加減速

Dir ***動作方向***

識別子	値	内容
MNET_DIR_PLUS	0	プラス方向
MNET_DIR_MINUS	1	マイナス方向

備考

速度データはMNETM1SetSpeed関数で設定ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

例　スタートモード　：　FL定速

[VisualC++]

```
MNETM1EscEL(0, 3, MNET_STAMODE_FL, MNET_ DIR_PLUS);
```

[VisualBasic]

```
MNETM1EscEL 0, 3, MNET_STAMODE_FL, MNET_ DIR_PLUS
```

[VB.NET]

```
MNETM1EscEL(0, 3, MNET_STAMODE_FL, MNET_ DIR_PLUS)
```

ライン0，デバイス3に動作方向：プラス，スタートモード:FL定速でEL(エンドリミット)からの脱出移動指令を発行します。
(マイナスエンドリミットからのプラス方向移動となります)

6－1－1－9. MNETM1Ez

機能

EZ信号カウント動作を起動します。
(スタートコマンドを発行します)
指定した回数のEZ信号をカウントするまでの動作となります。

書式

[VisualC++]

```
short __stdcall MNETM1Ez(
      short lineNo,
      short devNo,
      short startMode,
      short dir);
```

[VisualBasic]

```
Declare Function MNETM1Ez Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal startMode As Integer,
      ByVal dir As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM1Ez Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal startMode As Short,
      ByVal dir As Integer) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

startMode ***スタートモード***

識別子	値	内容
MNET_STAMODE_FL	0050h	FL定速
MNET_STAMODE_FH	0051h	FH 定速
MNET_STAMODE_UD	0053h	加減速

Dir ***動作方向***

識別子	値	内容
MNET_DIR_PLUS	0	プラス方向
MNET_DIR_MINUS	1	マイナス方向

備考

EZカウント回数は事前にMNETM1SetPar関数で設定した値が適用されます。
加減速動作の場合はEZカウント完了により減速停止となります。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

例　スタートモード ： FL定速

[VisualC++]

```
MNETM1Ez(0, 3, MNET_STAMODE_FL, MNET_ DIR_PLUS);
```

[VisualBasic]

```
MNETM1Ez 0, 3, MNET_STAMODE_FL, MNET_ DIR_PLUS
```

[VB.NET]

```
MNETM1Ez(0, 3, MNET_STAMODE_FL, MNET_ DIR_PLUS)
```

ライン0, デバイス3に動作方向:プラス, スタートモード:FL定速でEZカウント移動指令を発行します。
(EZカウント回数は事前にMNETM1SetPar関数で設定した値が適用されます)

6－1－1－10. MNETM1Stop

機能

軸を停止します。
(ストップコマンドを発行します)

書式

[VisualC++]

```
short __stdcall MNETM1Stop(
      short lineNo,
      short devNo,
      short stopMode);
```

[VisualBasic]

```
Declare Function MNETM1EscEL Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal stopMode As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM1EscEL Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal stopMode As Short) As Integer
```

パラメータ

lineNo ***ライン番号*** (0始まり)
devNo ***デバイス番号*** (0始まり)
stopMode ***停止モード指定***

識別子	値	内容
MNET_STOP	0	即停止
MNET_SDSTOP	1	減速停止
MNET_EMGSTOP	2	非常停止

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

[VisualC++]

```
MNETM1Stop(0, 3, MNET_STOP);
```

[VisualBasic]

```
MNETM1Stop 0, 3, MNET_STOP
```

[VB.NET]

```
MNETM1Stop(0, 3, MNET_STOP)
```

ライン0, デバイス3に即停止コマンドを発行します。

6－1－1－11. MNETM1GetSpeed

機能

速度データ取得

書式

[VisualC++]

```
short __stdcall MNETM1GetSpeed(
      short lineNo,
      short devNo,
      double* speed);
```

[VisualBasic]

```
Declare Function MNETM1GetSpeed Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByRef speed As Double) As Integer
```

[VB.NET]

```
Declare Function MNETM1GetSpeed Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByRef speed As Double) As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
speed	***取得データ格納変数***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照）

使用例

[VisualC++]

```
double speed;

MNETM1GetSpeed(0, 3, &speed);
```

[VisualBasic]

```
Dim speed As Double

MNETM1GetSpeed 0, 3, speed
```

[VB.NET]

```
Dim speed As Double

MNETM1GetSpeed(0, 3, speed)
```

ライン0, デバイス3の速度データを変数speedに取得します。

6－1－1－12. MNETM1GetCounter

機能

カウンタ値取得

書式

[VisualC++]

```
short __stdcall MNETM1GetCounter(
        short lineNo,
        short devNo,
        short selCnt,
        long* data);
```

[VisualBasic]

```
Declare Function MNETM1GetCounter Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal selCnt As Integer,
        ByRef data As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM1GetCounter Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal selCnt As Short,
        ByRef data As Integer) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

selCnt ***対象カウンタ選択***

識別子	値	内容
MNET_SEL_COUNTER1	0	カウンタ1
MNET_SEL_COUNTER2	1	カウンタ2
MNET_SEL_COUNTER3	2	カウンタ3

data ***取得データ格納変数***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

[VisualC++]

```
long data;

MNETM1GetCounter(0, 3, MNET_SEL_COUNTER1, &data);
```

[VisualBasic]

```
Dim data As Long

MNETM1GetCounter 0, 3, MNET_SEL_COUNTER1, data
```

[VB.NET]

```
Dim data As Integer

MNETM1GetCounter(0, 3, MNET_SEL_COUNTER1, data)
```

ライン0，デバイス3のカウンタ1のデータを変数dataに取得します。

6－1－1－13. MNETM1SetCounter

機能

カウンタ値設定

書式

[VisualC++]

```
short __stdcall MNETM1SetCounter(
        short lineNo,
        short devNo,
        short selCnt,
        long data);
```

[VisualBasic]

```
Declare Function MNETM1SetCounter Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal selCnt As Integer,
        ByVal data As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM1SetCounter Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal selCnt As Short,
        ByVal data As Integer) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

selCnt ***対象カウンタ選択***

識別子	値	内容
MNET_SEL_COUNTER1	0	カウンタ1
MNET_SEL_COUNTER2	1	カウンタ2
MNET_SEL_COUNTER3	2	カウンタ3

data ***設定値***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

[VisualC++]

```
MNETM1SetCounter(0, 3, MNET_SEL_COUNTER1, 200);
```

[VisualBasic]

```
MNETM1SetCounter 0, 3, MNET_SEL_COUNTER1, 200
```

[VB.NET]

```
MNETM1SetCounter(0, 3, MNET_SEL_COUNTER1, 200)
```

ライン0, デバイス3のカウンタ1にデータ:200を設定します。

Motionnet® Remote I/O & Remote Motion

6－1－1－14. MNETM1GetLtc

機能

ラッチデータ取得

書式

[VisualC++]

```
short __stdcall MNETM1GetLtc(
      short lineNo,
      short devNo,
      short sel,
      long* data);
```

[VisualBasic]

```
Declare Function MNETM1GetLtc Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal sel As Integer,
      ByRef data As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM1GetLtc Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal sel As Short,
      ByRef data As Short) As Integer
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

sel ***対象ラッチ選択***

識別子	値	内容
MNET_SEL_LTC1	0	ラッチ1
MNET_SEL_LTC2	1	ラッチ2
MNET_SEL_LTC3	2	ラッチ3

data ***取得データ格納変数***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

[VisualC++]

```
long data;

MNETM1GetLtc(0, 3, MNET_SEL_LTC1, &data);
```

[VisualBasic]

```
Dim data As Long

MNETM1GetLtc 0, 3, MNET_SEL_LTC1, data
```

[VB.NET]

```
Dim data As Integer

MNETM1GetLtc(0, 3, MNET_SEL_LTC1, data)
```

ライン0, デバイス3のラッチ1のデータを変数dataに取得します。

6－1－1－15. MNETM1GetBitDataStruct

機能

ビットデータ読出し(構造体に一括)

書式

[VisualC++]

```
short __stdcall MNETM1GetBitDataStruct(
      short lineNo,
      short devNo,
      TMNETM1BitData*  bit);
```

[VisualBasic]

```
Declare Function MNETM1GetBitDataStruct Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByRef bit As TMNETM1BitData) As Integer
```

[VB.NET]

```
Declare Function MNETM1GetBitDataStruct Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByRef bit As TMNETM1BitData) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
bit	***ビットデータ構造体*** (6－3－1. ビットデータ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

[VisualC++]

```
TMNETM1BitData bit = {0};       // ビットデータ構造体定義

MNETM1GetBitDataStruct(0, 3, &bit);   // ライン0, デバイス3のビットデータを一括取得
If (bit.spel != 0)              // spel(プラスEL)ビットによる判定
  m_pel.SetCheck(TRUE);         // ON時:チェックON
ELse
  m_pel.SetCheck(FALSE);        // OFF時: チェックOFF

If (bit.salm  != 0)             // salm(ALM)ビットによる判定
  m_alm.SetCheck(TRUE);         // ON時: チェックON
ELse
  m_alm.SetCheck(FALSE);        // OFF時: チェックOFF
```

[VisualBasic]

```
Dim bit As TMNETM1BitData       ’ビットデータ構造体定義

MNETM1GetBitDataStruct 0, 3, bit  ’ライン0, デバイス3のビットデータを一括取得
If (bit.spel <> 0) Then         ’spel(プラスEL)ビットによる判定
  ShpPEL.FillColor = &HFF&      ’ON時:明赤色
ELse
  ShpPEL.FillColor = &H80&      ’OFF時:暗赤色
End If

If (bit.salm <> 0) Then         ’salm(ALM)ビットによる判定
  ShpAlm.FillColor = &HFF&      ’ON時:明赤色
ELse
  ShpAlm.FillColor = &H80&      ’OFF時:暗赤色
End If
```

[VB.NET]

```
Dim bit As TMNETM1BitData       ’ビットデータ構造体定義

MNETM1GetBitDataStruct(0, 3, bit)  ’ライン0, デバイス3のビットデータを一括取得
If (bit.spel <> 0) Then         ’  spel(プラスEL)ビットによる判定
  ShpPEL.BackColor = System.Drawing.Color.Red  ’  ON時:明赤色
ELse
  ShpPEL.BackColor = System.Drawing.Color.DarkRed       ’OFF時:暗赤色
End If

If (bit.salm <> 0) Then         ’  salm(ALM)ビットによる判定
  ShpAlm.BackColor = System.Drawing.Color.Red  ’  ON時:明赤色
ELse
  ShpAlm.BackColor = System.Drawing.Color.DarkRed       ’  OFF時:暗赤色
End If
```


Motionnet®
Remote I/O & Remote Motion

6－1－1－16. MNETM1GetBitData

機能

ビットデータ取得

書式

[VisualC++]

```
short __stdcall MNETM1GetBitData(
      short lineNo,
      short devNo,
      short sel,
      short* data);
```

[VisualBasic]

```
Declare Function MNETM1GetBitData Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal sel As Integer,
      ByRef data As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM1GetBitData Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal sel As Short,
      ByRef data As Short) As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
sel	***対象選択*** （6－3－3－9. ビット選択参照）
data	***取得データ格納変数***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照）

使用例

[VisualC++]

```
short data;

MNETM1GetBitData(0, 3, MNETM1_SEL_BIT_MPOUT1, &data);
```

[VisualBasic]

```
Dim data As Integer

MNETM1GetBitData 0, 3, MNETM1_SEL_BIT_MPOUT1, data
```

[VB.NET]

```
Dim data As Short

MNETM1GetBitData(0, 3, MNETM1_SEL_BIT_MPOUT1, data)
```

ライン0, デバイス3のビットMPOUT1の状態を変数dataに取得します。

6－1－1－17. MNETM1SetBitData

機能

ビットデータ設定

書式

[VisualC++]

```
short __stdcall MNETM1SetBitData(
        short lineNo,
        short devNo,
        short sel,
        short data);
```

[VisualBasic]

```
Declare Function MNETM1SetBitData Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal sel As Integer,
        ByVal data As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM1SetBitData Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal sel As Short,
        ByVal data As Short) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
sel	***対象選択*** (6－3－3－9. ビット選択参照) Set可能なビット以外は指定できません。
data	***設定値*** 1: ON, 0:OFF, -1:反転(ERCの場合は指定不可)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

[VisualC++]

```
MNETM1SetBitData(0, 3, MNETM1_SEL_BIT_MPOUT1, 1);
```

[VisualBasic]

```
MNETM1SetBitData 0, 3, MNETM1_SEL_BIT_MPOUT1, 1
```

[VB.NET]

```
MNETM1SetBitData(0, 3, MNETM1_SEL_BIT_MPOUT1, 1)
```

ライン0, デバイス3のビットMPOUT1に1を設定します。

6－1－1－18. MNETM1FixSpeedRange

機能

速度倍率固定

書式

[VisualC++]

```
short __stdcall MNETM1FixSpeedRange(
      short lineNo,
      short devNo,
      unsigned char range);
```

[VisualBasic]

```
Declare Function MNETM1FixSpeedRangELib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal range As Byte) As Integer
```

[VB.NET]

```
Declare Function MNETM1FixSpeedRangELib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal range As Byte) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
range	***倍率指定*** (6－3－3－8. 速度倍率レンジ参照)

備考

本関数は速度オーバーライドを行う際などに通常速度設定時に毎回自動選択される速度倍率を固定するために使用されます。ご利用にあたっては(6－2－10－1. 速度倍率)の項を参照ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

[VisualC++]

```
MNETM1FixSpeedRange(0, 3, MNETM1_SPEED_1_RANGE);
```

[VisualBasic]

```
MNETM1FixSpeedRange 0, 3, MNETM1_SPEED_1_RANGE
```

[VB.NET]

```
MNETM1FixSpeedRange(0, 3, MNETM1_SPEED_1_RANGE)
```

ライン0, デバイス3の速度倍率を1倍のレンジに固定します。

6－1－1－19. MNETM1UnFixSpeedRange

機能

速度倍率固定解除

書式

[VisualC++]

```
short __stdcall MNETM1UnFixSpeedRange(
      short lineNo,
      short devNo);
```

[VisualBasic]

```
Declare Function MNETM1UnFixSpeedRangELib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM1UnFixSpeedRangELib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short) As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照）

使用例

[VisualC++]

```
MNETM1UnFixSpeedRange(0, 3);
```

[VisualBasic]

```
MNETM1UnFixSpeedRange 0, 3
```

[VB.NET]

```
MNETM1UnFixSpeedRange(0, 3)
```

ライン0, デバイス3の速度倍率の固定を解除します。

6－1－1－20. MNETM1OutCsta

機能

CSTA信号出力

書式

[VisualC++]

```
short __stdcall MNETM1OutCsta(
      short lineNo,
      short devNo);
```

[VisualBasic]

```
Declare Function MNETM1OutCsta Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM1OutCsta Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short) As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）

備考

本関数はCSTA信号を外部出力できるI/Fを持ったモーションデバイスユニットにおいてのみ利用できます。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照）

使用例

[VisualC++]

```
MNETM1OutCsta(0, 3);
```

[VisualBasic]

```
MNETM1OutCsta 0, 3
```

[VB.NET]

```
MNETM1OutCsta(0, 3)
```

ライン0, デバイス3のCSTA信号を出力します。

6－1－1－21. MNETM1OutCstp

機能

CSTP信号出力

書式

[VisualC++]

```
short __stdcall MNETM1OutCstp(
      short lineNo,
      short devNo);
```

[VisualBasic]

```
Declare Function MNETM1OutCstp Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM1OutCstp Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short) As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）

備考

本関数はCSTP信号を外部出力できるI/Fを持ったモーションデバイスユニットにおいてのみ利用できます。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照）

使用例

[VisualC++]

```
MNETM1OutCstp(0, 3);
```

[VisualBasic]

```
MNETM1OutCstp 0, 3
```

[VB.NET]

```
MNETM1OutCstp(0, 3)
```

ライン0, デバイス3のCSTP信号を出力します。

6－1－1－22. MNETM1OvrideSpeed

機能

速度オーバーライド

書式

[VisualC++]

```
short __stdcall MNETM1OvrideSpeed(
      short lineNo,
      short devNo,
      double newSpeed);
```

[VisualBasic]

```
Declare Function MNETM1OvrideSpeed Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal newSpeed As Double) As Integer
```

[VB.NET]

```
Declare Function MNETM1OvrideSpeed Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal newSpeed As Double) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
newSpeed	***設定速度***

備考

・指令受付時に指定速度まで達する十分な残りパルスがない場合はエラーとなります。
(オーバーライドされません)
・位置決め動作関数(MNETM1Inc,MNETM1Abs)によるスタートで、残りパルスがない際どい状況で
使用されますと、スローダウンポイントの設定が正しく行われない場合があります。
・指定できる速度は速度倍率設定により異なります。ご利用にあたっては(6－2－10－1. 速度倍率)の項を参照ください。
・本関数は加減速スタートの動作にのみ有効です。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

[VisualC++]

```
MNETM1OvrideSpeed(0, 3, 30000);
```

[VisualBasic]

```
MNETM1OvrideSpeed 0, 3, 30000
```

[VB.NET]

```
MNETM1OvrideSpeed(0, 3, 30000)
```

ライン0, デバイス3の速度を30000PPSに変更します。

6－1－1－23. MNETM1OvridePos

機能

目標位置オーバーライド

書式

[VisualC++]

```
short __stdcall MNETM1OvridePos(
      short lineNo,
      short devNo,
      long newPos);
```

[VisualBasic]

```
Declare Function MNETM1OvridePos Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal newPos As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM1OvridePos Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal newPos As Integer) As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
newPos	***設定値***

備考,注意

オーバーライドの成功失敗は本関数実行後 MNETM1GetREST 関数でオーバーライド失敗のビットの ON/OFF を確認ください。エラー割り込み要因データはオーバーライド失敗ビット以外のエラー割り込み要因ビットも含め、MNETM1GetREST 関数の実行によりリセットされますのでご使用においては注意ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照）

使用例

[VisualC++]

```
long rest;

MNETM1OvridePos(0, 3, 10000);
MNETGetREST(0, 3, &rest);
```

[VisualBasic]

```
Dim rest As Long

MNETM1OvridePos 0, 3, 10000
MNETGetREST 0, 3, rest
```

[VB.NET]

```
Dim rest As Integer

MNETM1OvridePos(0, 3, 10000)
MNETGetREST(0, 3, rest)
```

ライン0, デバイス3の目標位置(移動量)を10000に変更します。
オーバーライドに成功していれば変数restに格納されたエラー割り込み要因データの12Bit目はOFFとなります。 (6－3－2－2. エラー割り込み要因参照)

6－1－1－24. MNETM1ResetSEND

機能

SEND(位置決め完了)ビットリセット

書式

[VisualC++]

```
short __stdcall MNETM1ResetSEND(
      short lineNo,
      short devNo);
```

[VisualBasic]

```
Declare Function MNETM1ResetSEND Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM1ResetSEND Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)

備考

・ご利用に当たっては(6－3－2－1. メインステータス)、および(6－2－1－3. 停止確認)の項を参照ください。

・MNETM1Inc 他の起動関数内部では、起動コマンド発行前に SEND ビットリセット処理が行なわれております。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

[VisualC++]

```
MNETM1ResetSEND(0, 3);
```

[VisualBasic]

```
MNETM1ResetSEND 0, 3
```

[VB.NET]

```
MNETM1ResetSEND(0, 3)
```

ライン0, デバイス3のSENDビットをリセットします。

Motionnet® Remote I/O & Remote Motion

6－1－1－25. MNETM1SetPar

機能

パラメータ設定

書式

[VisualC++]

```
short __stdcall MNETM1SetPar(
      short lineNo,
      short devNo,
      short itemNo,
      long val);
```

[VisualBasic]

```
Declare Function MNETM1SetPar Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal itemNo As Integer,
      ByVal val As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM1SetPar Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal itemNo As Short,
      ByVal val As Integer) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
itemNo	***項目選択*** (6－3－3－10. パラメータ項目番号定義参照)
val	***設定値***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

[VisualC++]

```
MNETM1SetPar(0, 3, MNETM1_PAR_OUTPLS_TYPE, 0);
```

[VisualBasic]

```
MNETM1SetPar 0, 3, MNETM1_PAR_OUTPLS_TYPE, 0
```

[VB.NET]

```
MNETM1SetPar(0, 3, MNETM1_PAR_OUTPLS_TYPE, 0)
```

ライン0, デバイス3のパラメータ「パルスタイプ」を0にセットします。

6－1－1－26. MNETM1GetPar

機能

パラメータ取得

書式

[VisualC++]

```
short __stdcall MNETM1GetPar(
      short lineNo,
      short devNo,
      short itemNo,
      long* val);
```

[VisualBasic]

```
Declare Function MNETM1GetPar Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal itemNo As Integer,
      ByRef val As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM1GetPar Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal itemNo As Short,
      ByRef val As Integer) As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
itemNo	***項目選択***
	（6－3－3－10. パラメータ項目番号定義参照）
val	***取得データ格納変数***。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照）

使用例

[VisualC++]

```
long data;

MNETM1GetPar(0, 3, MNETM1_PAR_OUTPLS_TYPE, &data);
```

[VisualBasic]

```
Dim data As Long

MNETM1GetPar 0, 3, MNETM1_PAR_OUTPLS_TYPE, data
```

[VB.NET]

```
Dim data As Integer

MNETM1GetPar(0, 3, MNETM1_PAR_OUTPLS_TYPE, data)
```

ライン 0, デバイス 3 のパラメータ「パルスタイプ」の設定値を変数 data に取得します。

6－1－1－27. MNETM1SearchORG

機能

原点サーチ移動
(スタートコマンドを発行します)

書式

[VisualC++]

```
short __stdcall MNETM1SearchORG(
      short lineNo,
      short devNo,
      short startMode,
      short dir,
      long amount);
```

[VisualBasic]

```
Declare Function MNETM1SearchORG Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal startMode As Integer,
      ByVal dir As Integer,
      ByVal amount As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM1SearchORG Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal startMode As Short,
      ByVal dir As Short,
      ByVal amount As Integer) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

startMode ***スタートモード***

識別子	値	内容
MNET_STAMODE_FL	0050h	FL定速
MNET_STAMODE_FH	0051h	FH 定速
MNET_STAMODE_UD	0053h	加減速

Dir ***動作方向***

識別子	値	内容
MNET_DIR_PLUS	0	プラス方向
MNET_DIR_MINUS	1	マイナス方向

amount ***原点抜け出し移動量***

正数を設定ください。

注意

減速時間が加速時間の 2 倍以上となる加減速度設定の場合は利用できません。

備考

原点復帰モードなどはMNETM1SetPar関数で設定します。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照）

使用例

例　スタートモード　：　FL定速

[VisualC++]

```
MNETM1SearchORG(0, 3, MNET_STAMODE_FL, MNET_DIR_MINUS, 1000);
```

[VisualBasic]

```
MNETM1SearchORG 0, 3, MNET_STAMODE_FL, MNET_DIR_MINUS, 1000
```

[VB.NET]

```
MNETM1SearchORG(0, 3, MNET_STAMODE_FL, MNET_DIR_MINUS, 1000)
```

ライン0, デバイス3に動作方向: マイナス, スタートモード:FL定速,
ORG抜け出し移動量:1000パルスで原点サーチ移動指令を発行します。

6－1－1－28. MNETM1GetREST

機能

エラー割り込み要因取得

書式

[VisualC++]

```
short __stdcall MNETM1GetREST(
      short lineNo,
      short devNo,
      long* data);
```

[VisualBasic]

```
Declare Function MNETM1GetREST Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByRef data As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM1GetREST Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByRef data As Integer) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
data	***取得データ格納変数***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

注意

このデータはモーションデバイスLSI:G9003のRESTレジスタ(エラー割り込みステータス)から取得しています。 このレジスタは読出しによりリセットされますのでご注意ください。

備考

ご利用にあたっては(6－3－2－2. エラー割り込み要因)の項を参照ください。

使用例

[VisualC++]

```
long data;

MNETM1GetREST(0, 3, &data);
```

[VisualBasic]

```
Dim data As Long

MNETM1GetREST 0, 3, data
```

[VB.NET]

```
Dim data As Integer

MNETM1GetREST(0, 3, data)
```

ライン0のデバイス3のエラー割り込み要因データを変数dataに取得します。

6－1－1－29. MNETM1GetRIST

機能

イベント割り込み要因取得

書式

[VisualC++]

```
short __stdcall MNETM1GetRIST(
      short lineNo,
      short devNo,
      long* data);
```

[VisualBasic]

```
Declare Function MNETM1GetRIST Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByRef data As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM1GetRIST Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByRef data As Integer) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
data	***取得データ格納変数***

注意

このデータはモーションデバイスLSI:G9003のRISTレジスタ(イベント割り込みステータス)から取得しています。 このレジスタは読出しによりリセットされますのでご注意ください。

備考

イベントの設定はMNETM1SetPar関数で行います。
ご利用にあたっては(6－3－2－3. イベント割り込み要因)の項を参照ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

[VisualC++]

```
long data;

MNETM1GetRIST(0, 3, &data);
```

[VisualBasic]

```
Dim data As Long

MNETM1GetRIST 0, 3, data
```

[VB.NET]

```
Dim data As Integer

MNETM1GetRIST(0, 3, data)
```

ライン0のデバイス3のイベント割り込み要因データを変数dataに取得します。

6－1－1－30. MNETM1Restart

機能

位置決めリスタート
(リスタートコマンドを発行します)

書式

[VisualC++]

```
short __stdcall MNETM1Restart(
      short lineNo,
      short devNo,
      short startMode);
```

[VisualBasic]

```
Declare Function MNETM1Restart Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal startMode As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM1Restart Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal startMode As Short) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)
devNo ***デバイス番号*** (0始まり)
startMode ***スタートモード***

識別子	値	内容
MNET_STAMODE_FL	0050h	FL定速
MNET_STAMODE_FH	0051h	FH 定速
MNET_STAMODE_UD	0053h	加減速

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

備考

ご利用にあたっては(6－2－8. リスタート)の項を参照ください。

使用例

例　スタートモード　：　FL定速

[VisualC++]

```
MNETM1Restart(0, 3, MNET_STAMODE_FL);
```

[VisualBasic]

```
MNETM1Restart 0, 3, MNET_STAMODE_FL
```

[VB.NET]

```
MNETM1Restart(0, 3, MNET_STAMODE_FL)
```

ライン0, デバイス3にスタートモード:FL定速でリスタート指令を発行します。

6－1－1－31. MNETM1SoftPCS

機能

PCS入力代行コマンド発行

書式

[VisualC++]

```
short __stdcall MNETM1SoftPCS(
      short lineNo,
      short devNo);
```

[VisualBasic]

```
Declare Function MNETM1SoftPCS Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM1SoftPCS Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)

備考

ご利用にあたっては(6－2－12. PCS 入力)の項を参照ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

[VisualC++]

```
MNETM1SoftPCS(0, 3);
```

[VisualBasic]

```
MNETM1SoftPCS 0, 3
```

[VB.NET]

```
MNETM1SoftPCS(0, 3)
```

ライン0, デバイス3にPCS入力代行コマンドを発行します。

6－1－1－32. MNETM1SoftLTC

機能

LTC入力代行コマンド発行

書式

[VisualC++]

```
short __stdcall MNETM1SoftLTC(
      short lineNo,
      short devNo);
```

[VisualBasic]

```
Declare Function MNETM1SoftLTC Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM1SoftLTC Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short) As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）

備考

ご利用にあたっては(6－2－19. ラッチ)の項を参照ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照）

使用例

[VisualC++]

```
MNETM1SoftLTC(0, 3);
```

[VisualBasic]

```
MNETM1SoftLTC 0, 3
```

[VB.NET]

```
MNETM1SoftLTC(0, 3)
```

ライン0, デバイス3にLTC入力代行コマンドを発行します。

6－1－1－33. MNETM1Reset

機能

モーションデバイスリセット

書式

[VisualC++]

```
short __stdcall MNETM1Reset(
      short lineNo,
      short devNo);
```

[VisualBasic]

```
Declare Function MNETM1Reset Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM1Reset Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short) As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）

備考

本関数の実行により、対象のモーションデバイス LSI:G9003 のレジスタはリセットされます。
したがって対象のデバイスに設定したパラメータなどは再設定を行う必要があります。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照）

使用例

[VisualC++]

```
MNETM1Reset (0, 3);
```

[VisualBasic]

```
MNETM1Reset 0, 3
```

[VB.NET]

```
MNETM1Reset(0, 3)
```

ライン0, デバイス3のモーションデバイスをリセットします。

Motionnet® Remote I/O & Remote Motion

6－1－1－34. MNETM1SetDLLPar

機能

DLL内部パラメータ設定

書式

[VisualC++]

```
short __stdcall MNETM1SetDLLPar(
      short lineNo,
      short devNo,
      short itemNo,
      char val);
```

[VisualBasic]

```
Declare Function MNETM1SetDLLPar Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal itemNo As Integer,
      ByVal val As Byte) As Integer
```

[VB.NET]

```
Declare Function MNETM1SetDLLPar Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal itemNo As Short,
      ByVal val As Byte) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
itemNo	***項目選択*** (6－3－3－11. DLL内部パラメータ項目番号定義) 参照
val	***設定値***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

[VisualC++]

```
MNETSetDLLPar(0, 0, MNETM1_DLLPAR_PREMOVE_CHK, 1);
```

[VisualBasic]

```
MNETSetDLLPar 0, 0, MNETM1_DLLPAR_PREMOVE_CHK, 1
```

[VB.NET]

```
MNETSetDLLPar(0, 0, MNETM1_DLLPAR_PREMOVE_CHK, 1)
```

DLL内部パラメータ「軸動作前確認省略」を1(省略する)にセットします。

6－1－1－35. MNETM1ReadParFile

機能

パラメータファイル読み出し
指定されたパラメータファイルを読み出し、読み出した値をレジスタに設定します。

書式

[Visual C++]

```
short __stdcall MNETM1ReadParFile(
                    short lineNo,
                    short devNo,
                    const char* filename);
```

[Visual Basic]

```
Declare Function MNETM1ReadParFile Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal filename As String)
                          As Integer
```

[VB.NET]

```
Declare Function MNETM1ReadParFile Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal filename As String)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
filename	***ファイル名***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照）

使用例

[Visual C++]

```
MNETM1ReadParFile(0, 0, “ABC.par”);
```

[Visual Basic]

```
MNETM1ReadParFile 0, 0, “ABC.par”
```

[VB.NET]

```
MNETM1ReadParFile(0, 0, “ABC.par”)
```

パラメータファイル”ABC.par”を読み出し、ライン0、デバイス0に設定します。

6－1－1－36. MNETM1WriteParFile

機能

パラメータファイル書き込み
指定されたファイルに環境設定などのレジスタデータを書き込みます。

書式

[Visual C++]

```
short __stdcall MNETM1WriteParFile(
                    short lineNo,
                    short devNo,
                    const char* filename);
```

[Visual Basic]

```
Declare Function MNETM1WriteParFile Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal filename As String)
                          As Integer
```

[VB.NET]

```
Declare Function MNETM1WriteParFile Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal filename As String)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
filename	***ファイル名***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照)

使用例

[Visual C++]

```
MNETM1WriteParFile(0, 0, “XYZ.txt”);
```

[Visual Basic]

```
MNETM1WriteParFile 0, 0, “XYZ.txt”
```

[VB.NET]

```
MNETM1WriteParFile(0, 0, “XYZ.txt”)
```

環境設定などのレジスタデータをファイル”XYZ.txt”に書き込みます。

6－1－2. ローカル１軸モーションデバイス基本関数

6－1－2－1. MNETM1WriteRegister

機能

モーションデバイスレジスタ書込み

書式

[VisualC++]

```
short __stdcall MNETM1WriteRegister(
      short lineNo,
      short devNo,
      unsigned short cmd,
      long data);
```

[VisualBasic]

```
Declare Function MNETM1WriteRegister Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal cmd As Integer,
      ByVal data As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM1WriteRegister Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal cmd As Short,
      ByVal data As Integer) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

cmd ***書き込みコマンド***

モーションデバイスLSI:G9003のレジスタ書込みコマンドを指定します。

data ***書込みデータ***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照）

注意

(4. 基本関数)の項を十分に理解頂いた上でご利用ください。

使用例

[VisualC++]

```
MNETM1WriteRegister(0, 3, WRCMP3, 1000);
```

[VisualBasic]

```
MNETM1WriteRegister 0, 3, WRCMP3, 1000
```

[VB.NET]

```
MNETM1WriteRegister(0, 3, WRCMP3, 1000)
```

ライン0のデバイス3のRCMP3レジスタにデータ1000を書込みます。

6－1－2－2. MNETM1ReadRegister

機能

モーションデバイスレジスタ読出し

書式

[VisualC++]

```
short __stdcall MNETM1ReadRegister(
        short lineNo,
        short devNo,
        unsigned short cmd,
        long* data);
```

[VisualBasic]

```
Declare Function MNETM1ReadRegister Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal cmd As Integer,
        ByRef data As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM1ReadRegister Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal cmd As Short,
        ByRef data As Integer) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
cmd	***書き込みコマンド***
	モーションデバイスLSI:G9003のレジスタ読出しコマンドを指定します。
data	***取得データ格納変数***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照）

注意

(4. 基本関数)の項を十分に理解頂いた上でご利用ください。

使用例

[VisualC++]

```
long data;

MNETM1ReadRegister (0, 3, RRCMP3, &data);
```

[VisualBasic]

```
Dim data As Long

MNETM1ReadRegister 0, 3, RRCMP3, data
```

[VB.NET]

```
Dim data As Integer

MNETM1ReadRegister(0, 3, RRCMP3, data)
```

ライン0のデバイス3のRCMP3レジスタのデータを変数dataに取得します。

6－1－2－3. MNETM1WriteCommand

機能

モーションデバイスコマンド書込み

書式

[VisualC++]

```
short __stdcall MNETM1WriteCommand(
      short lineNo,
      short devNo,
      unsigned short cmd);
```

[VisualBasic]

```
Declare Function MNETM1WriteCommand Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal cmd As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM1WriteCommand Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal cmd As Short) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
cmd	***書き込みコマンド***
	モーションデバイスLSI:G9003のコマンドを指定します。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 6－3－4. 戻り値参照）

注意

(4. 基本関数)の項を十分に理解頂いた上でご利用ください。

使用例

[VisualC++]

```
MNETM1WriteCommand(0, 3, STOP);
```

[VisualBasic]

```
MNETM1WriteCommand 0, 3, STOP
```

[VB.NET]

```
MNETM1WriteCommand(0, 3, STOP)
```

ライン0のデバイス3にSTOPコマンドを発行します。

6－2. 機能操作説明

6－2－1. データ取得

6－2－1－1. 状態取得

1 軸モーションデバイスの入出力信号状態、ステータスビット、および軸動作状態は以下の２つの関数を使用し取得可能です。

[MNETM1GetBitData 関数]

対象ビット(データ)を指定し、その項目のデータを取得します。

記述例

[VisualC++]

```
short data;

MNETM1GetBitData(0, 3, MNETM1_SEL_BIT_MPOUT1, &data);
```

[VisualBasic]

```
Dim data As Integer

MNETM1GetBitData 0, 3, MNETM1_SEL_BIT_MPOUT1, data
```

[VB.NET]

```
Dim data As Short

MNETM1GetBitData(0, 3, MNETM1_SEL_BIT_MPOUT1, data)
```

ライン0のデバイス3のビットMPOUT1の状態を変数dataに取得します。

[MNETM1GetBitDataStruct 関数]

ビットデータ構造体にビット(データ)を一括で取得します。

使用例

[VisualC++]

```
TMNETM1BitData bit = {0};       // ビットデータ構造体定義

MNETM1GetBitDataStruct(0, 3, &bit);   // ライン0のデバイス3のビットデータを一括取得
If (bit.spel != 0)              // spel (プラスEL)ビットによる判定
  m_pel.SetCheck(TRUE);         // ON時:チェックON
ELse
  m_pel.SetCheck(FALSE);        // OFF時: チェックOFF

If (bit.salm  != 0)             // salm(ALM)ビットによる判定
  m_alm.SetCheck(TRUE);         // ON時: チェックON
ELse
  m_alm.SetCheck(FALSE);        // OFF時: チェックOFF
```

[VisualBasic]

```
Dim bit As TMNETM1BitData       ’ビットデータ構造体定義

MNETM1GetBitDataStruct 0, 3, bit  ’ライン0のデバイス3のビットデータを一括取得
If (bit.pel <> 0) Then          ’spel (プラスEL)ビットによる判定
  ShpPEL.FillColor = &HFF&      ’ON時:明赤色
ELse
  ShpPEL.FillColor = &H80&      ’OFF時:暗赤色
End If

If (bit.alm <> 0) Then          ’salm(ALM)ビットによる判定
  ShpAlm.FillColor = &HFF&      ’ON時:明赤色
ELse
  ShpAlm.FillColor = &H80&      ’OFF時:暗赤色
End If
```

[VB.NET]

```
Dim bit As TMNETM1BitData       ’ビットデータ構造体定義

MNETM1GetBitDataStruct(0, 3, bit)  ’ライン0のデバイス3のビットデータを一括取得
If (bit.pel <> 0) Then          ’spel (プラスEL)ビットによる判定
  ShpPEL.BackColor = System.Drawing.Color.Red  ’ON時:明赤色
ELse
  ShpPEL.BackColor = System.Drawing.Color.DarkRed       ’OFF時:暗赤色
End If

If (bit.alm <> 0) Then          ’salm(ALM)ビットによる判定
  ShpAlm.BackColor = System.Drawing.Color.Red  ’ON時:明赤色
ELse
  ShpAlm.BackColor = System.Drawing.Color.DarkRed       ’OFF時:暗赤色
End If
```


Motionnet®
Remote I/O & Remote Motion

6−2−1−2. カウンタ取得

1軸モーションデバイスのカウンタデータの取得は、MNETM1GetCounter 関数で行います。
また各カウンタに接続(設定)する信号の設定は MNETM1SetPar 関数で行います。

記述例

[VisualC++]

```
long data;

MNETM1GetCounter(0, 3, MNET_SEL_COUNTER1, &data);
```

[VisualBasic]

```
Dim data As Long

MNETM1GetCounter 0, 3, MNET_SEL_COUNTER1, data
```

[VB.NET]

```
Dim data As Integer

MNETM1GetCounter(0, 3, MNET_SEL_COUNTER1, data)
```

ライン0のデバイス3のカウンタ1のデータを変数dataに取得します。

6−2−1−3. 停止確認

MNETM1GetBitDataStruct 関数、あるいは MNETM1GetBitData 関数を使用して以下のビットやデータの取得を行い判定できます。

SEND

動作停止により割り込みを発生し、1になります。割り込みリセットコマンド(MNETResetSEND関数実行)で0になります。

SERR

エラー割り込み発生により 1 になり、REST レジスタ読み出し(MNETM1GetREST 関数実行)で 0 になります。

SEVT

イベント割り込み発生により1になり、RISTレジスタ読み出し(MNETM1GetRIST関数実行)で0になります。

以上をまとめると SEND を取得することで動作停止の判定が行え、あわせて SERR で正常停止であるのかエラー発生による停止であるのかを判定できます。
また「cnd」ビット(データ)の取得により軸の動作状態(加速中/減速中ほか)の確認が可能です。

入力ポート変化割込の設定

また 1 軸モーションデバイス PORT0 の入力ポート変化割込設定機能を有効にすることで、入力ポート変化割込フラグでの動作停止、割込発生を確認できます。
ただし 1 軸モーションデバイス PORT0 の入力ポート変化割込機能は通常の入力ポート変化割込と動作が異なる点があり注意が必要です。

①入力変化の監視は BIT0(SINT)だけが対象です。また入力変化割込が ON となるのは BIT0 の立上り変化時となっています。
②BIT0(SINT)は BIT1(SEND), BIT2(SERR), BIT3(SEVT)の OR 信号であるため、BIT1～3 すべてを OFF にしないと入力変化割込の監視対象である BIT0 は OFF となりません。

以上をまとめると 1 軸モーションデバイスの停止確認に入力ポート変化割込を使用する場合、入力ポート変化割込が ON 時、まず入力ポート変化フラグをクリアし、その後変化割込の基となっている SINT を OFF にする(SEND,SERR,SEVT を OFF にする)処理が必要となります。

6−2−1−4. エラー割り込み要因取得

エラー割り込み要因の取得は MNETM1GetREST 関数を使用します。

このデータ(取得元であるモーションデバイス LSI:G9003 の REST レジスタ)は読出しによりリセットされます。

6−2−1−5. イベント割り込み要因取得

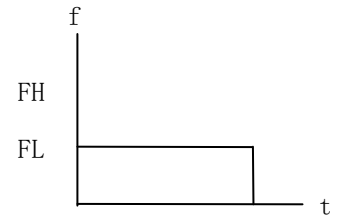
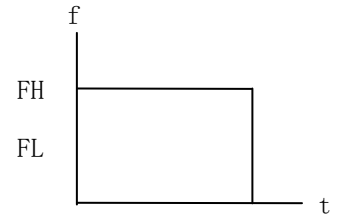
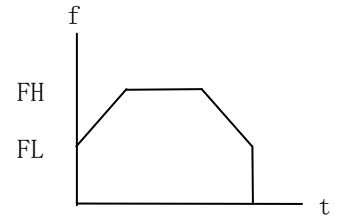
イベント割り込み要因の取得は MNETM1GetRIST 関数を使用します。

このデータ(取得元であるモーションデバイス LSI:G9003 の REST レジスタ)は読出しによりリセットされます。また、イベント割込の設定は MNETM1SetPar 関数で行います。

6−2−2. スタートモード

スタートモードは下記3パターンがあります。

MNETM1SetSpeed関数での速度データ設定時、速度データ構造体のメンバー「StartMode」に、また移動命令関数の引数に設定するデータとその識別子は次のようになります。

識別子	値	内容
MNET_STAMODE_FL	0050h	FL定速  速度データ構造体メンバー「SpdL」速度での定速動作となります。
MNET_STAMODE_FH	0051h	FH定速  速度データ構造体メンバー「SpdH」速度での定速動作となります。
MNET_STAMODE_UD	0053h	加減速  速度データ構造体メンバー「SpdL」速度から「SpdH」速度に加速、定速移動し、減速して停止します。 加速度はメンバー「Acc」、減速度は「Dec」で設定します。

Motionnet®
Remote I/O & Remote Motion

6−2−3. 速度データ構造体

MNETM1SetSpeed 関数での速度データの設定に使用します。

書式

[VisualC++]

```
struct TMNETM1Cmd {
  double SpdL;                  // 移動速度(FL)
  double SpdH;                  // 移動速度(FH)
  double SpdA;                  // 移動速度(FA)
  double Acc;                   // 加速データ
  double Dec;                   // 減速データ
  double SAccSect;              // 加速S字区間
  double SDecSect;              // 減速S字区間
  short StartMode;              // スタートモード(FL定速/FH定速/加減速)
  short AdcMtd;                 // 加減速度指定方法(レート/時間)
  short SMode;                  // S字モード(S字無し/S字有り)
};
```

[VisualBasic]

```
Type TMNETM1Cmd
  SpdL As Double      '/ 移動速度(FL)
  SpdH As Double      '/ 移動速度(FH)
  SpdA As Double      '/ 移動速度(FA)
  Acc AS Double       '/ 加速データ
  Dec AS Double       '/ 減速データ
  SAccSect As Double  '/ 加速 S 字区間
  SDecSect As Double  '/ 減速 S 字区間
  StartMode As Integer'/ スタートモード(FL 定速/FH 定速/加減速)
  AdcMtd As Integer   '/ 加減速度指定方法(レート/時間)
  SMode As Integer    '/ S 字モード(S 字無し/S 字有り)
End Type
```

[VB.NET]

```
Structure TMNETM1Cmd
  Dim SpdL As Double      '/ 移動速度(FL)
  Dim SpdH As Double      '/ 移動速度(FH)
  Dim SpdA As Double      '/ 移動速度(FA)
  Dim Acc As Double       '/ 加速データ(ms/ PPS/S)
  Dim Dec As Double       '/ 減速データ(ms/ PPS/S)
  Dim SAccSect As Double '/ 加速S字区間
  Dim SDecSect As Double '/ 減速S字区間
  Dim StartMode As Short '/ スタートモード(FL定速/FH定速/加減速)
  Dim AdcMtd As Short     '/ 加減速指定方法(レート/時間)
  Dim SMode As Short      '/ S字モード(S字無し/S字有り)
End Structure
```

[メンバー]

メンバー	内容	設定値	設定可能範囲
SpdL	FL速度	PPS	0.1～6.5MPPS(注1)
SpdH	FH速度	PPS	0.1～6.5MPPS(注1)
SpdA	FA速度	PPS	0.1～6.5MPPS(注1) (注4)
Acc	加速データ	AdcMtdの指定に応じ,加速レートあるいは加速時間として扱われます。	0以上のデータを設定(注2)
Dec	減速データ	AdcMtdの指定に応じ,減速レートあるいは減速時間として扱われます。	0以上のデータを設定(注2)
SAccSect	加速S字区間	設定値0の場合、直線部分のないS字動作となります	0～(SpdH-SpdL)/2(注3)
SDecSect	減速S字区間	設定値0の場合、直線部分のないS字動作となります	0～(SpdH-SpdL)/2(注3)
StartMode	スタートモード	50h ： FL定速 51h ： FH定速 53h ： 加減速	左記から選択
AdcMtd	加減速度指定方法	0 ： レート(1秒あたりの変化パルス速度) 1 ： 時間(秒)	左記から選択
Smode	S字モード	0 ： S字無し 1 ： S字有り	左記から選択

(注1) MNETM1FixSpeedRange関数で速度倍率固定を行った場合は設定した速度倍率によります。
詳細は(6－2－10－1. 速度倍率)の項を参照ください。

(注2) AdcMtdで設定した加減速度指定方法に従い、FL速度、FH速度、S字設定、加減速データからモーションデバイスLSI:G9003のレジスタ設定値が計算されます。
計算された値がレジスタの最大値/最小値を超えた場合にエラーとするか、最大値/最小値に自動補正するかはパラメータで選択できます。詳細は(6－2－4－2.加減速設定の自動補正)の項を参照ください。

(注3) (6－2－5. S字動作設定)の項を参照ください。

(注4) FA速度は原点復帰時の逆転定速動作およびバックラッシュ補正動作において適用されます。

MNETM1SetSpeed関数で設定されるデータはスタートモード別に以下のようになります。
また後述の「設定上の注意」を必ずご確認ください。

設定されるデータ

FL定速の場合

SpdL : FL速度 必ず設定が必要です。(速度0は速度設定エラーとなります)
SpdH : FH速度
SpdA : FA速度
StartMode : スタートモード 必ず設定が必要です。

上記以外のメンバーのデータは無視されます。 FH速度，FA速度の設定は必ず必要ではありません。

FH定速の場合

SpdL : FL速度
SpdH : FH速度 必ず設定が必要です。(速度0は速度設定エラーとなります)
SpdA : FA速度
StartMode : スタートモード 必ず設定が必要です。

上記以外のメンバーのデータは無視されます。 FL速度，FA速度の設定は必須ではありません。

加減速の場合

SpdL : FL速度 必ず設定が必要です。
SpdH : FH速度 必ず設定が必要です。(速度0は速度設定エラーとなります)
SpdA : FA速度
Acc : 加速データ 必ず設定が必要です。
Dec : 減速データ 必ず設定が必要です。
SaccSect : 加速S字範囲
SdecSect : 減速S字範囲
StartMode : スタートモード 必ず設定が必要です。
AdcMtd : 加減速指定方法 必ず設定が必要です。
SMode : S字使用/未使用 必ず設定が必要です。

FA速度、S字範囲の設定は必須ではありません。

設定上の注意

①VisualC++での使用の場合

構造体定義時にメンバーを下記のように初期化(0クリア)してください。

```
TMNETM1Cmd cmd = {0};   //構造体定義と初期化(0クリア)
```

初期化を行わない場合、メンバーに残った不定データにより正しい動作が行われない可能性があります。

(VisualBasic,VB.NETの場合は定義時に自動的に初期化されますので必要ありません)

②DLL 内部の速度計算について

SpdL/SpdH/SpdAに設定された速度は3つの速度の中に最も大きい速度を基準に、速度設定できる速度間隔が決まりモーションデバイスに設定する速度が演算されます。3つの速度の中に66.6pps以下の小さい速度を含む場合、最も大きい速度との関係で速度計算結果エラーとなる場合があります。

詳細は(6－2－10－1. 速度倍率)の項を参照ください。

③動作起動関数の引数「スタートモード」との関係

速度データ設定時のスタートモードが「FL 定速」あるいは「FH 定速」の場合、前述の通り加減速データは設定されません。(モーションデバイス内部の加減速設定(加減速レート)は前状態を維持します。(注))

動作起動関数(MNETM1Inc 他)の引数スタートモードにはMNETM1SetSpeed 関数での速度データ設定時のモードとは無関係に、任意のスタートモードを指定できますが、SD 信号入力による速度変更など加減速が行われる場合はあらかじめ、速度データ設定時にスタートモード「加減速」で加減速データの設定を行ってください。

(注) 速度データ設定時の加減速指定方法が「時間指定」であっても、維持されるのは速度データ設定時に DLL 内部で計算された加減速レートになります。

6−2−4. 加減速設定

6−2−4−1. 加減速指定方法

MNETM1SetSpeed関数での加減速データ設定時、加減速度を加速(減速)時間で指定するか、加速(減速)レート(1秒あたりの増分(減分)パルス速度)で指定するかを、速度データ構造体のメンバー「AdcMtd」に設定します。この指定に従い、メンバー「Acc」，「Dec」に設定された値は以下のデータとして扱われます。

識別子	値	内容
MNET_ADC_BY_RATE	0	レート指定(1秒あたりの増分(減分)パルス速度)
MNET_ADC_BY_TIME	1	時間指定　(秒)

6−2−4−2. 加減速設定の自動補正

設定された加減速データは DLL 内部で計算され、モーションデバイス LSI ： G9003 のレジスタに設定されますが計算された結果がレジスタ長を超える場合や設定可能最小値: 1 に満たない場合に設定可能最大値/最小値に自動補正する機能を DLL は持っています。この機能を有効にするかどうかはパラメータになっており、MNETSetDLLPar 関数を使用し設定できます。

識別子	値	内容	設定範囲	設定値／取得値 (網掛け：デフォルト)
MNETM1_DLLPAR_ADC_CORRECT	3E1h	加減速設定自動補正	0/1	0:補正する,1:補正しない

[補正する場合]

最大/最小値を超える場合は最大/最小値に補正されます。

[補正しない場合]

最大/最小値を超える場合は加速レート計算結果エラー/減速レート計算結果エラーになります。

6−2−5. S 字動作設定

S字動作設定はMNETM1SetSpeed関数での速度データ設定時に速度データ構造体メンバー「SMode」の設定で行います。

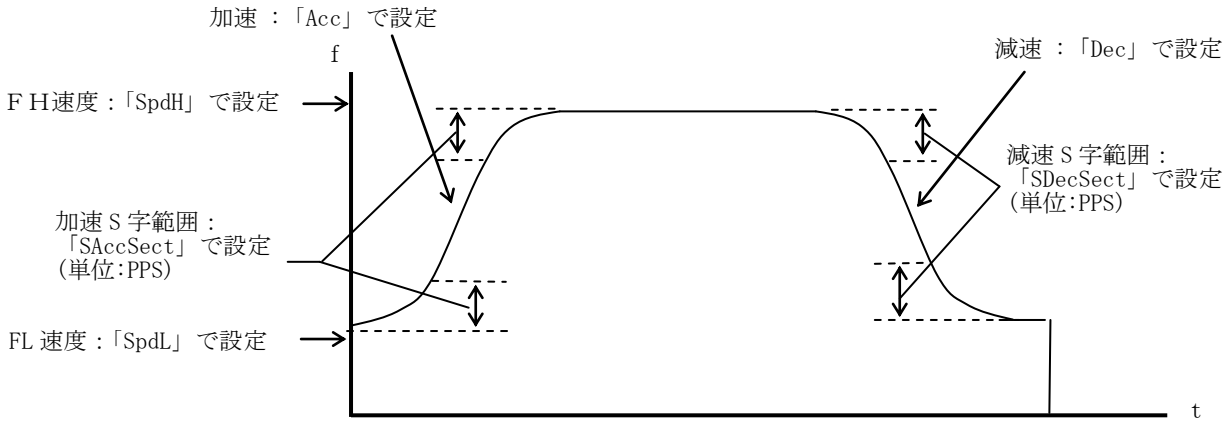
S 字動作の有り無しはメンバー「SMode」で行います。0 が S 字無し、1 が S 字有りとなります。

加減時における S 字動作の範囲はメンバー「SAccSect」に指定します。

減速時における S 字動作の範囲はメンバー「SDecSect」に指定します。

S 字有り（SMode：1）で加速 S 字範囲(「SAccSect」)に 0 を設定した場合、自動的に(SpdH-SpdL)/2 の値が適用され、加速全域にわたって S 字動作となり、直線加速部分のない加速となります。

同様に減速Ｓ字範囲(「SDecSect」)に 0 を設定した場合、自動的に(SpdH-SpdL)/2 の値が適用され、減速全域にわたって S 字動作となり、直線減速部分のない減速となります。



加減速動作と、対応する速度データ構造体メンバーの設定を下表にまとめます。

加減速動作	「SMode」設定	「SAccSect」，「SDecSect」設定
直線のみ加減速(全域が直線)	0	直線加減速となるため設定は無効です。
S字のみ加減速(全域がS字)	1	0
直線部分のあるS字加減速	1	1 以上 (SpdH-SpdL)/2 以下の範囲で設定します。(注)

(注) (SpdH-SpdL)/2 を超えた場合は(SpdH-SpdL)/2 に自動補正されます。

6−2−6. 原点復帰モード

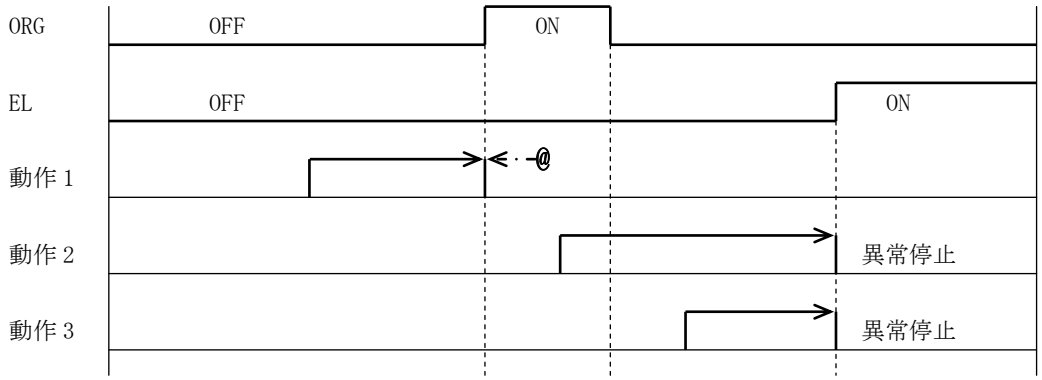
原点復帰動作の起動はMNETM1Org関数で行います。
原点復帰動作の設定はMNETM1SetPar関数で行います。

原点復帰動作においての注意など

① ソフトエンドリミットが設定されている場合は原点復帰動作においても有効となります。
② ORG信号を使用する原点復帰動作はORG入力のOFF→ONにより処理されます。
既にORG入力がONしている場合、あるいはORG信号センサーを越えた位置にいる場合、EL、あるいは設定したソフトエンドリミットまで動作しつづけることになりますのでご注意ください。
(MNETM1EscOrg関数、MNETM1ToEL関数などを使用することで原点復帰反対方向への脱出が行えます)
③ 原点復帰動作によりCOUNTER1～COUNTER3がリセットされます。
スタートモード/設定によりカウンタのリセット位置が異なりますのでご注意ください。

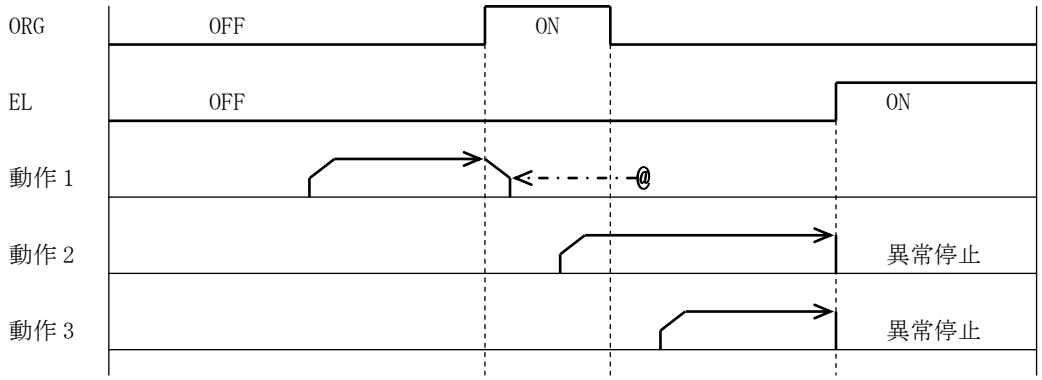
6−2−6−1. 原点復帰モード：0

□定速動作　＜センサ：EL, ORG＞



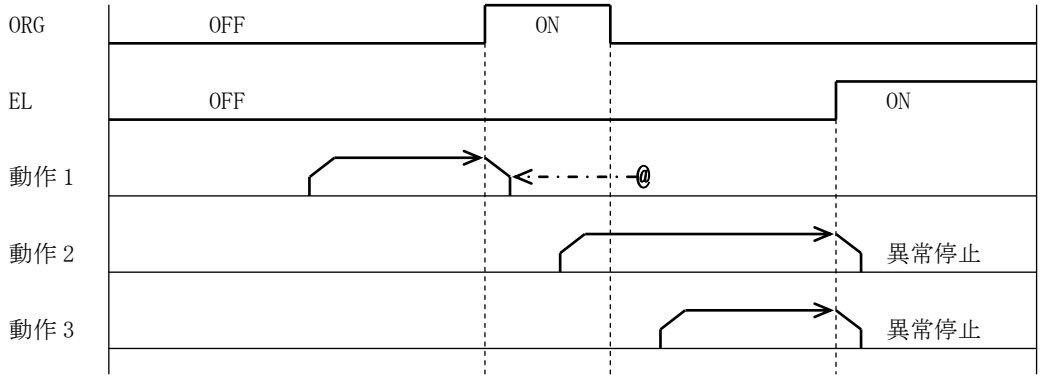
■加減速動作　＜センサ：EL, ORG＞

正常停止時でも原点位置ではありませんが、COUNTER2(機械位置)は信頼できる値となります。



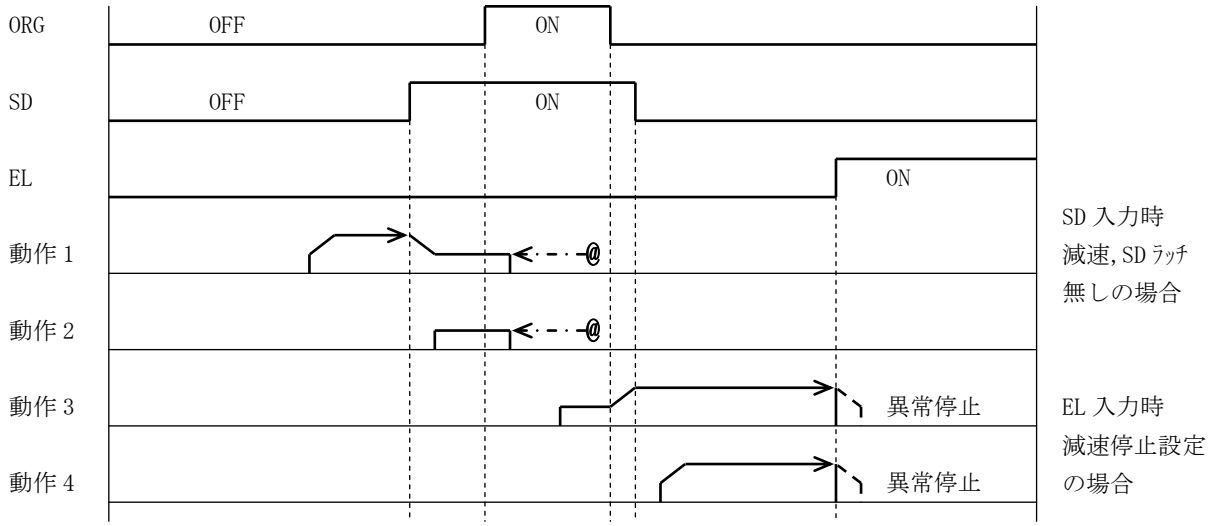
■加減速動作　＜センサ：EL, ORG＞

正常停止時でも原点位置ではありませんが、COUNTER2(機械位置)は信頼できる値となります。



Motionnet®
Remote I/O & Remote Motion

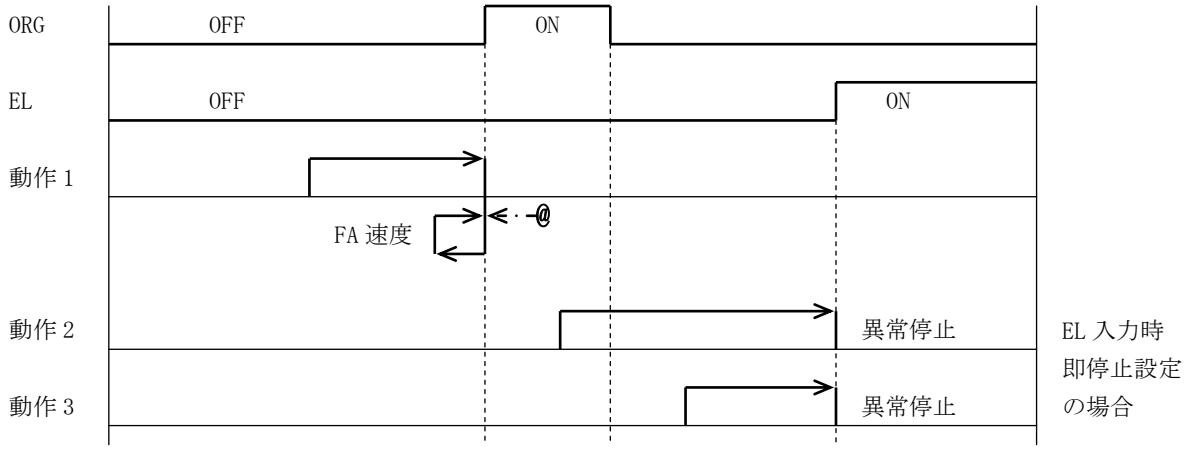
■加減速動作　＜センサ：EL, SD, ORG＞



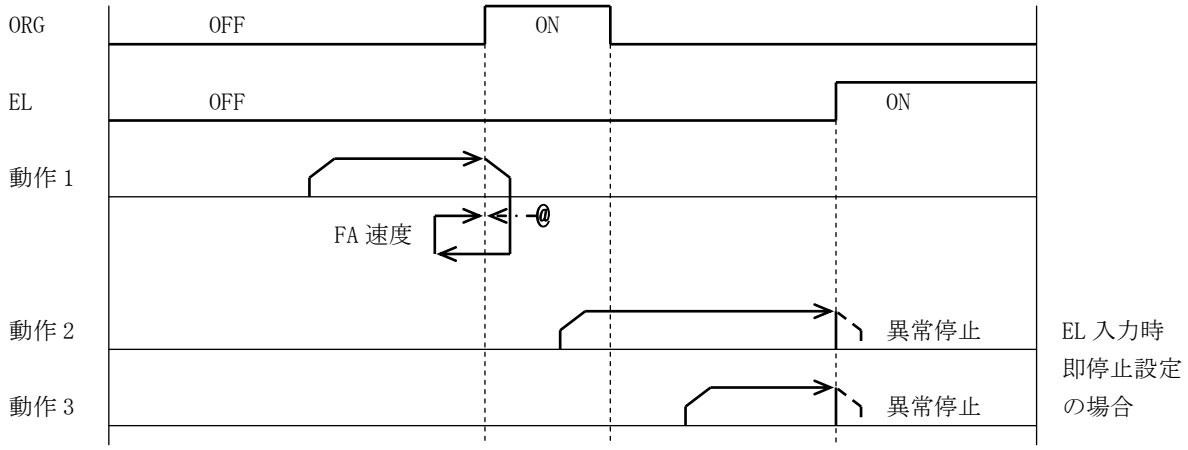
注意.@位置は原点復帰停止時に"ERC 信号を自動出力"に設定した時の ERC 信号出力のタイミングです。

6−2−6−2. 原点復帰モード：1

□定速動作　＜センサ：EL, ORG＞



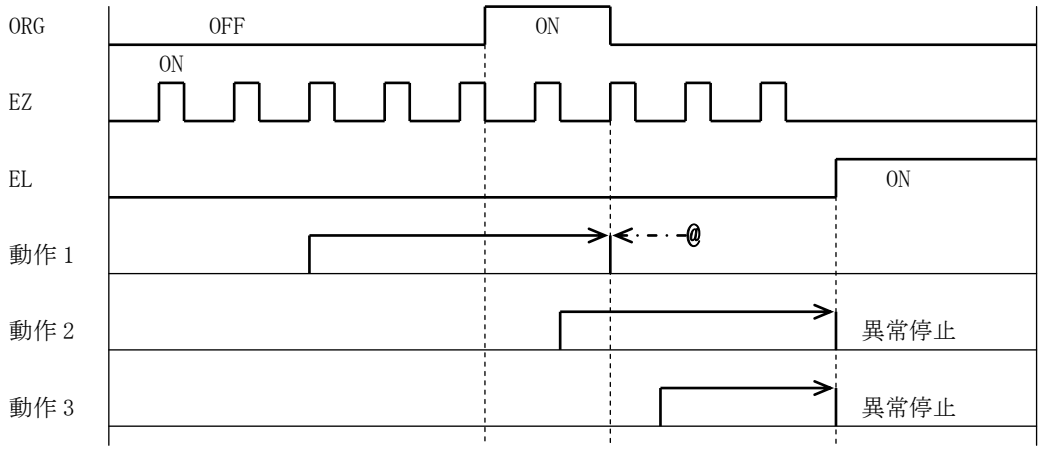
■加減速動作　＜センサ：EL, ORG＞



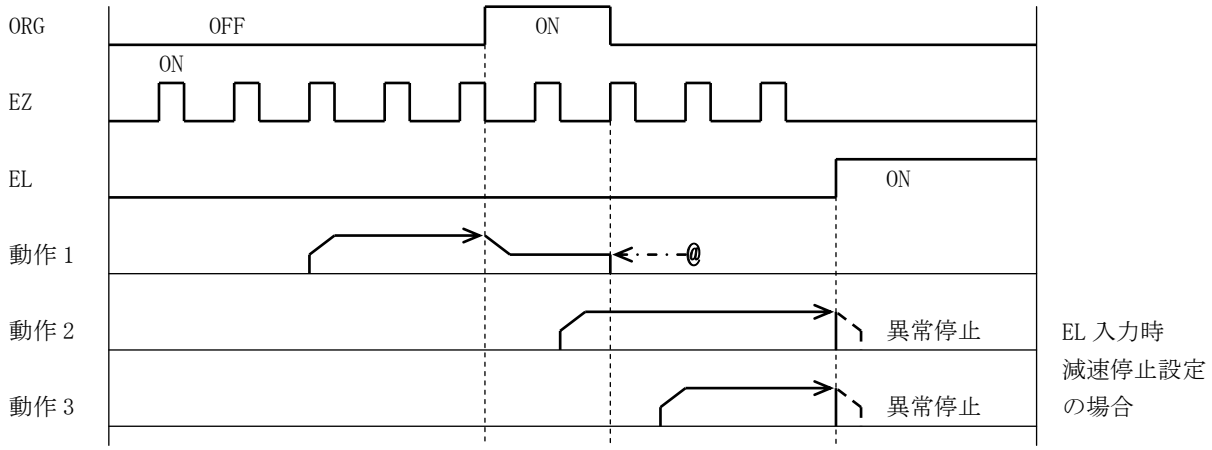
注意. @位置は原点復帰停止時に"ERC 信号を自動出力"に設定した時の ERC 信号出力のタイミングです。

6−2−6−3. 原点復帰モード：2

□定速動作　＜センサ：EL, ORG, EZ(Z 相回数 2 回目設定)＞



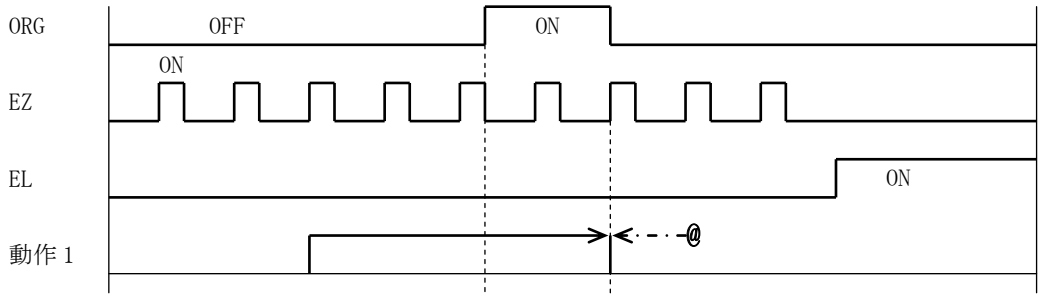
■加減速動作　＜センサ：EL, ORG, EZ(Z 相回数 2 回目設定)＞



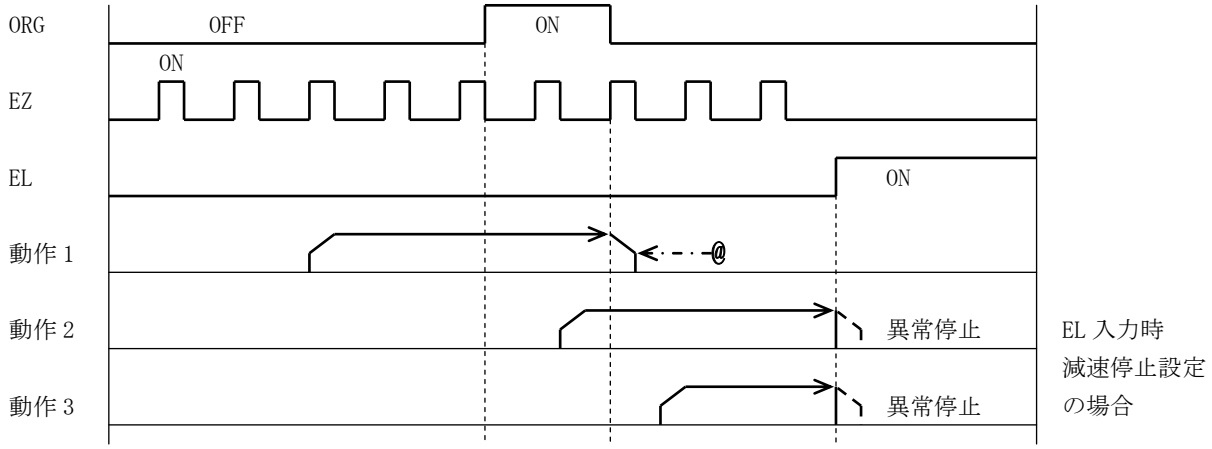
注意.@位置は原点復帰停止時に"ERC 信号を自動出力"に設定した時の ERC 信号出力のタイミングです。

6−2−6−4. 原点復帰モード：3

□定速動作　＜センサ：EL, ORG, EZ(Z 相回数 2 回目設定)＞



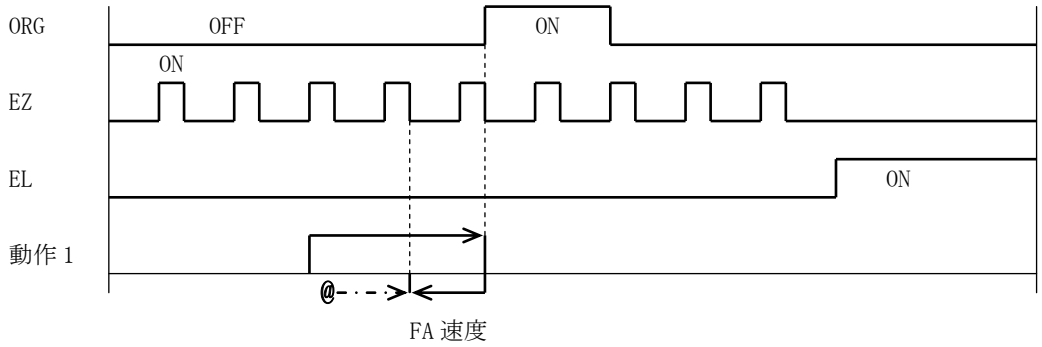
■加減速動作　＜センサ：EL, ORG, EZ(Z 相回数 2 回目設定)＞



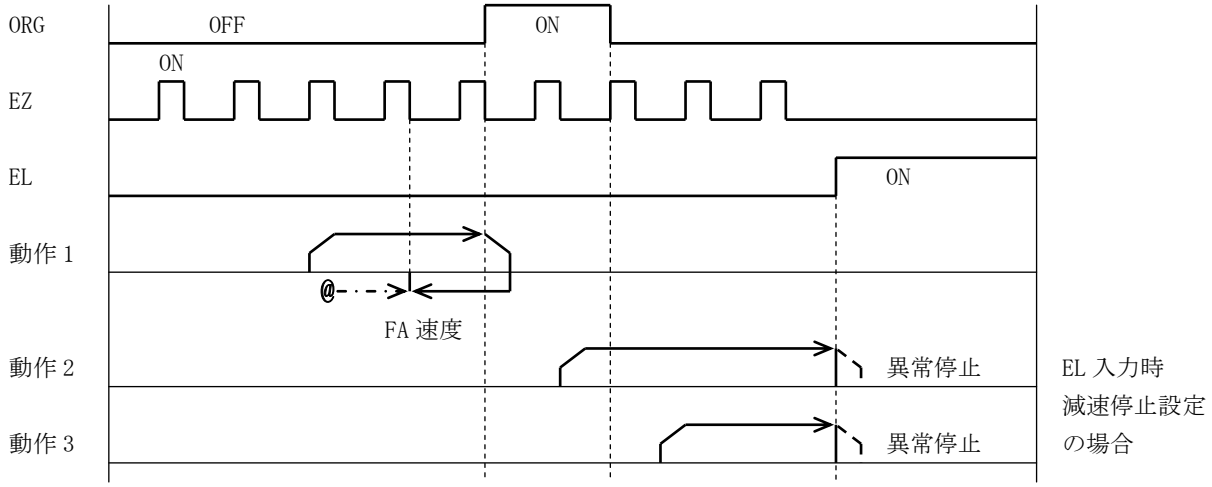
注意. @位置は原点復帰停止時に"ERC 信号を自動出力"に設定した時の ERC 信号出力のタイミングです。

6−2−6−5. 原点復帰モード：4

□定速動作　＜センサ：EL, ORG, EZ(Z 相回数 2 回目設定)＞



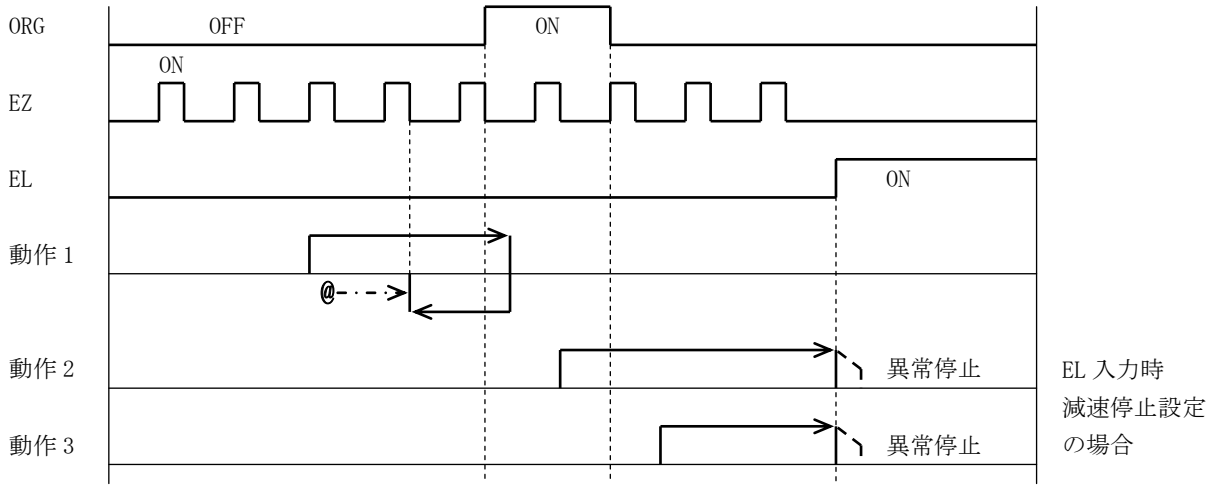
■加減速動作　＜センサ：EL, ORG, EZ(Z 相回数 2 回目設定)＞



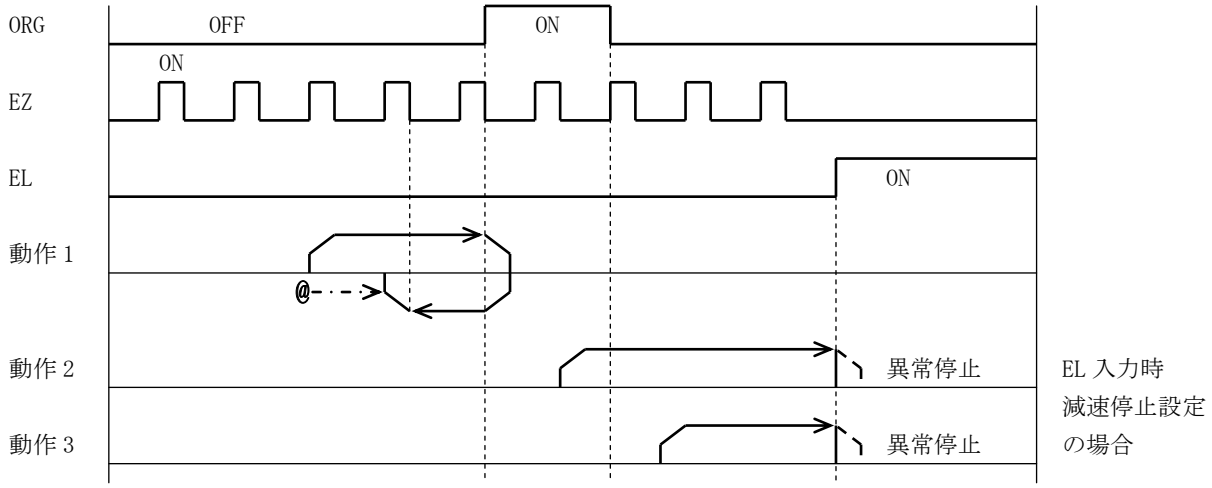
注意.@位置は原点復帰停止時に"ERC 信号を自動出力"に設定した時の ERC 信号出力のタイミングです。

6−2−6−6. 原点復帰モード：5

□定速動作　＜センサ：EL, ORG, EZ(Z 相回数 2 回目設定)＞



■加減速動作　＜センサ：EL, ORG, EZ(Z 相回数 2 回目設定)＞

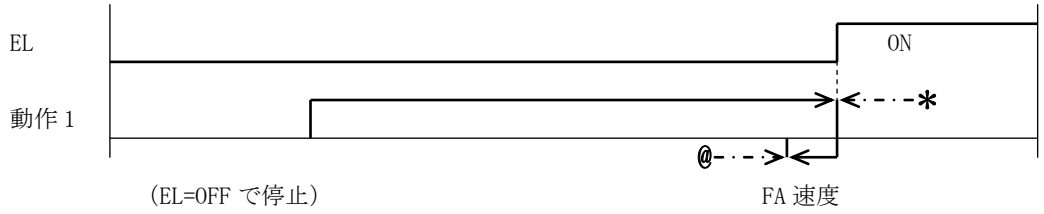


注意. @位置は原点復帰停止時に"ERC 信号を自動出力"に設定した時の ERC 信号出力のタイミングです。

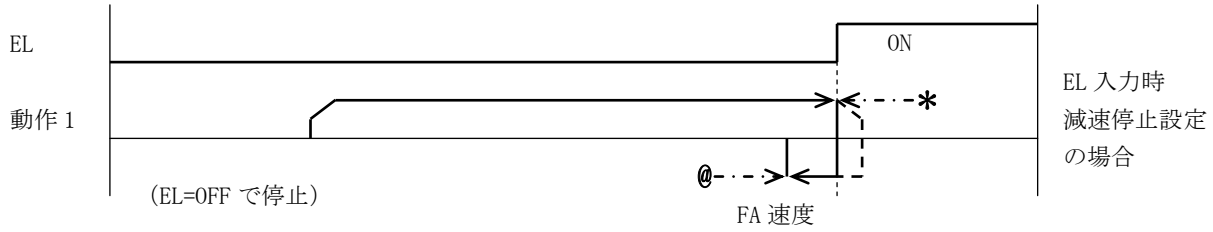
Motionnet®
Remote I/O & Remote Motion

6−2−6−7. 原点復帰モード：6

□定速動作　＜センサ：EL＞



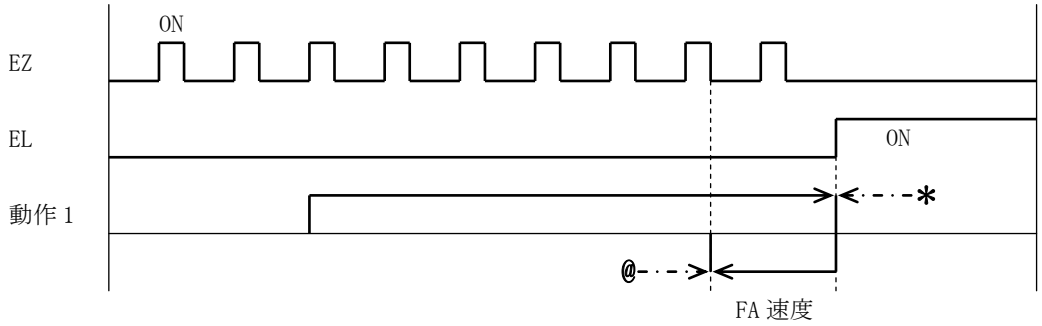
■加減速動作　＜センサ：EL＞



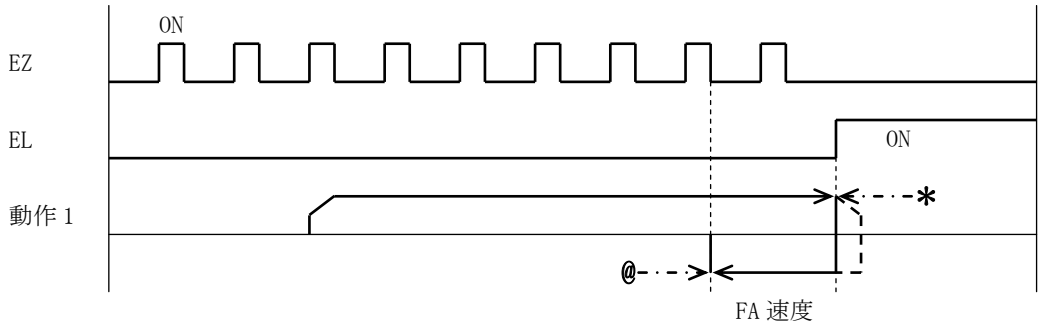
注意. @位置は原点復帰停止時に"ERC 信号を自動出力"に設定した時の ERC 信号出力のタイミングです。
またエラー停止時 ERC 出力 "する", かつ EL 入力時処理="即停止"の場合は、※の位置でも ERC 信号を出力します。

6−2−6−8. 原点復帰モード：7

□定速動作　＜センサ：EL，EZ(Z 相回数 2 回目設定)＞



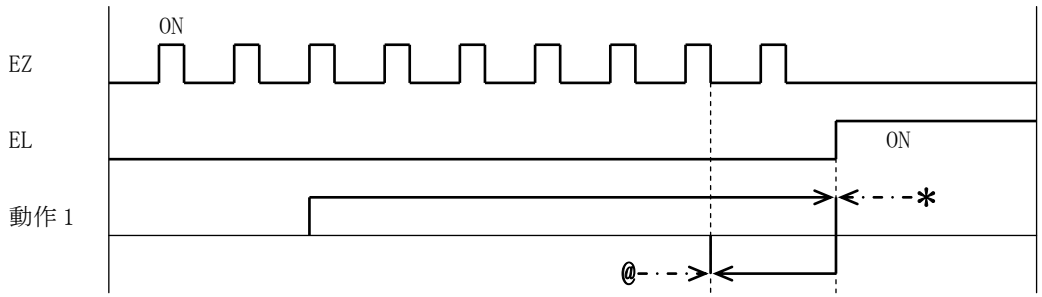
■加減速動作　＜センサ：EL，EZ(Z 相回数 2 回目設定)＞



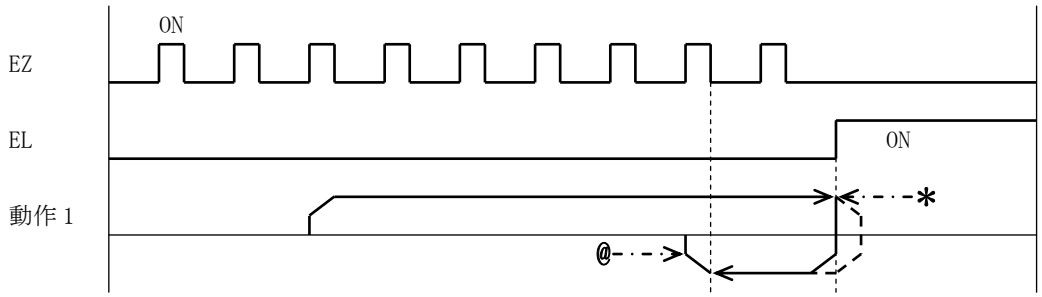
注意．@位置は原点復帰停止時に"ERC 信号を自動出力"に設定した時の ERC 信号出力のタイミングです。

6−2−6−9. 原点復帰モード：8

□定速動作　＜センサ：EL, EZ(Z 相回数 2 回目設定)＞

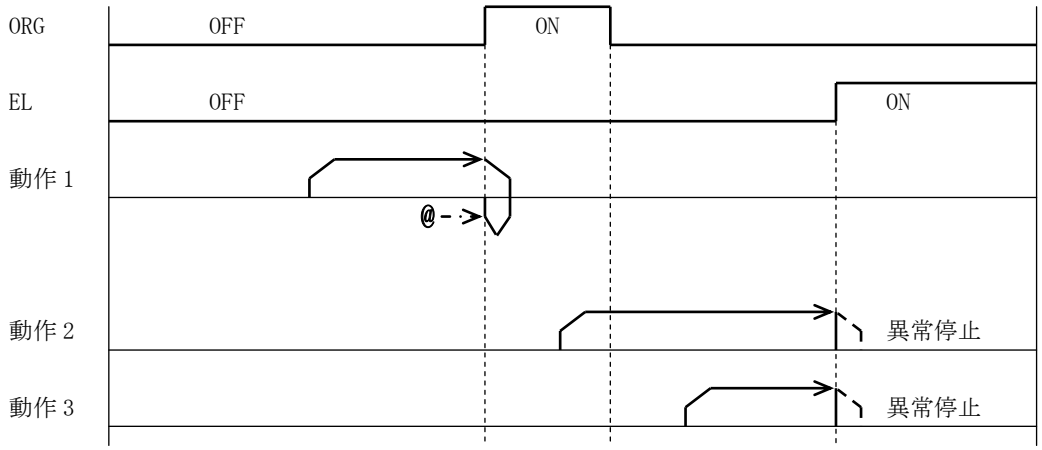


■加減速動作　＜センサ：EL, EZ(Z 相回数 2 回目設定)＞



6−2−6−10. 原点復帰モード：9

■加減速動作　＜センサ：EL, ORG)＞

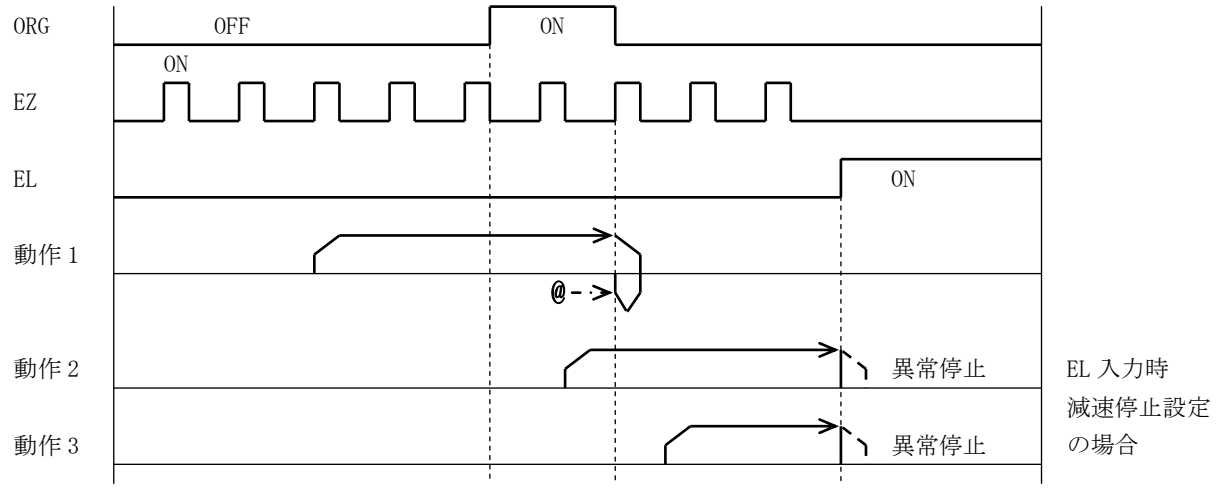


注意. @位置は原点復帰停止時に"ERC 信号を自動出力"に設定した時の ERC 信号出力のタイミングです。
またエラー停止時 ERC 出力 "する", かつ EL 入力時処理="即停止"の場合は、※の位置でも ERC 信号を出力します。

Motionnet®
Remote I/O & Remote Motion

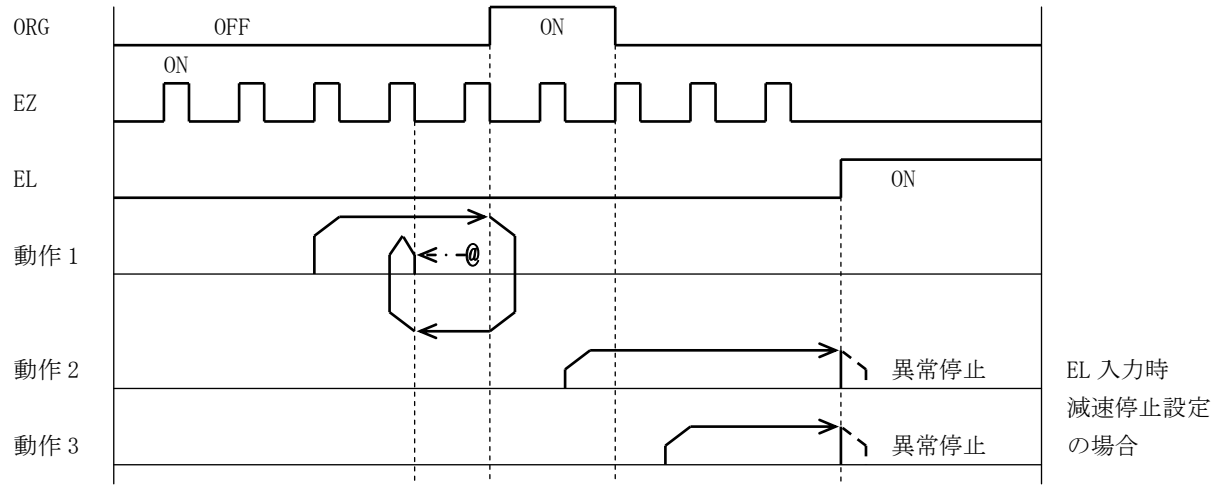
6−2−6−11. 原点復帰モード：10

■加減速動作 ＜センサ：EL, ORG, EZ(Z 相回数 2 回目設定)＞



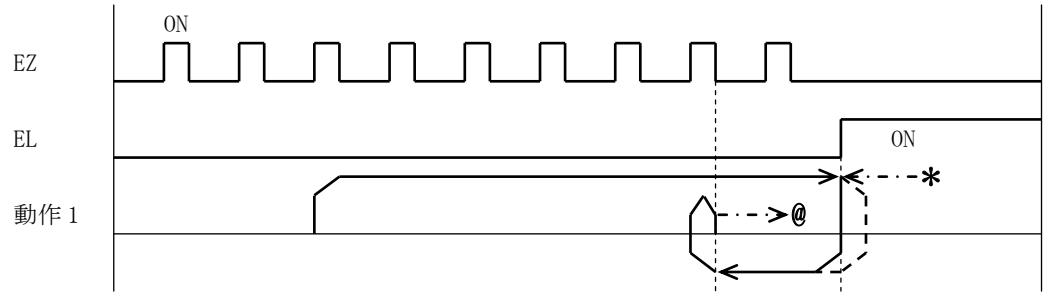
6−2−6−12. 原点復帰モード：11

■加減速動作 ＜センサ：EL, ORG, EZ(Z 相回数 2 回目設定)＞



6−2−6−13. 原点復帰モード：12

■加減速動作 ＜センサ：EL, EZ(Z 相回数 2 回目設定)＞



注意. @位置は原点復帰停止時に"ERC 信号を自動出力"に設定した時の ERC 信号出力のタイミングです。
またエラー停止時 ERC 出力 "する", かつ EL 入力時処理="即停止"の場合は、※の位置でも ERC 信号を出力します。

6−2−7. 原点サーチ動作

原点サーチ動作とは原点復帰動作に機能を追加したモードで、下記の動作ブロックで構成されます。

①指定と逆方向の「原点復帰動作」

②指定と逆方向の「位置決め動作による原点抜け出し動作」

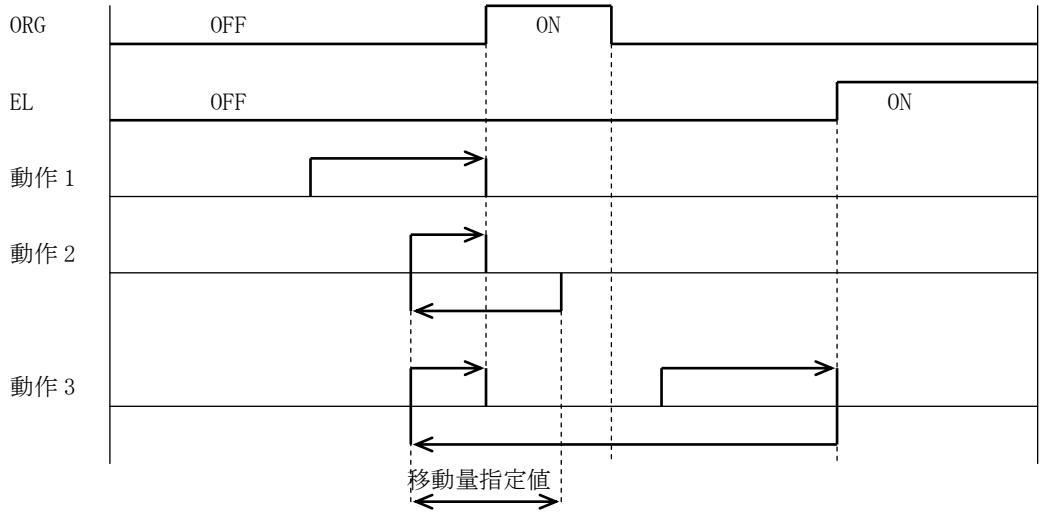
③指定方向の「原点復帰動作」

原点サーチ動作の起動はMNETM1SearchOrg関数で行います。

原点復帰モードはMNETM1SetPar関数で設定したモードとなります。(原点復帰モード:6,7,8,12では本関数は利用できません)

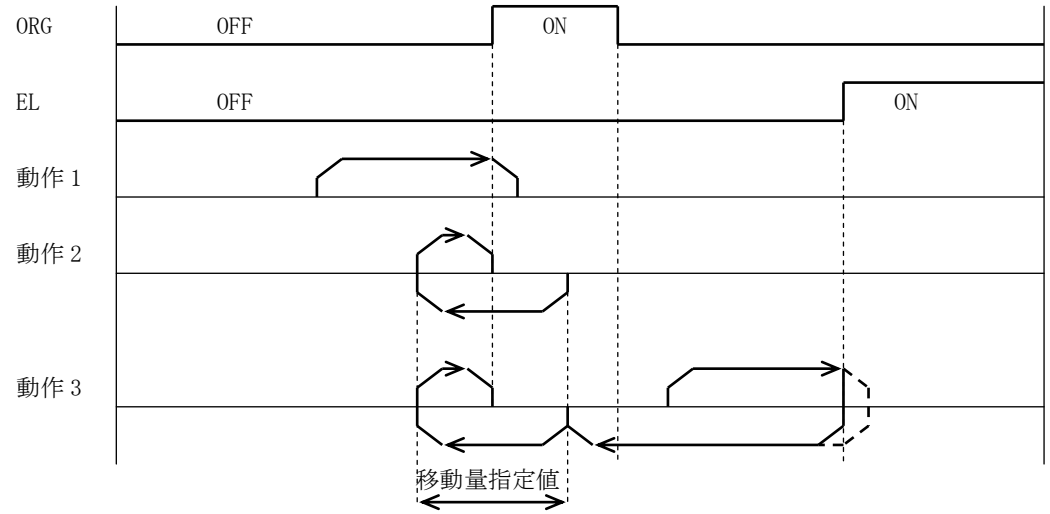
[原点復帰モード:0 の例]

□定速動作 ＜センサ：EL, ORG＞



■加減速動作 ＜センサ：EL, ORG＞

正常停止時でも原点位置ではありませんが、COUNTER2(機械位置)は信頼できる値となります。



動作１ ：スタート後ORG 入力がON で停止した時は正常停止になります。

動作２ ：スタート時に既にORG 入力がON している場合には、「位置決め動作による原点抜け出し動作」を行った後で、「原点復帰動作」を行います。

動作３ ：指定方向に動作してEL で停止した場合には、逆方向への「原点復帰動作」と「位置決め動作による原点抜け出し動作」を行った後で、再度、指定方向の「原点復帰動作」を行います。

尚、「位置決め動作による原点抜け出し動作」では、MNETM1SearchOrg関数で設定した移動量の位置決め動作が、原点を抜け出すまで繰り返し行われます。

Motionnet®
Remote I/O & Remote Motion

6−2−8. リスタート

位置決め動作をMNETM1Stop関数で中断した場合、MNETM1Restart関数を使用することで、当初の目標位置までの位置決め動作をリスタートすることができます。

使用する上での注意

①本関数はモーションデバイスLSI：G9003の残量パルス位置決め動作機能を利用しています。

②速度設定は停止前の動作と同じ速度設定となります。

6−2−9. 動作前確認

本DLLではモーション関数実行前の状態確認を行うか/省略するかはパラメータになっており、MNETSetDLLPar関数を使用し設定できます。

識別子	値	内容	設定範囲	設定値／取得値（網掛け：デフォルト）
MNETM1_DLLPAR_PREMOVE_CHK	3E0h	軸動作前確認省略	0/1	0:省略しない,1:省略

［省略しない場合］

「ライン番号」、「デバイス番号」、「デバイスがモーションデバイスかどうか」、「動作中でないこと」、「動作方向の EL 入力」、「ALM 入力」、「EMG 入力」、「位置決め指令値ソフトリミット確認」、「目的の状態の信号入力(ORG,EL など)」の確認が行われます。

ただし実行前に上記データ確認のための数回のデータ通信が発生します。

［省略する場合］

「ライン番号」、「デバイス番号」、「デバイスがモーションデバイスかどうか」、「動作中でないこと」の確認が行われますが、その他の入出力信号や指令値の確認などは行われません。

ALM などの入力がある場合、動作指令は発行後エラーとなり、エラー割り込みありビット(SERR)が ON することになります。エラー割り込み要因は MNETM1GetREST 関数で確認してください。

6−2−10. 速度設定に関する詳細事項

6−2−10−1. 速度倍率

本 DLL では通常の FL/FH/FA 速度設定範囲は 0.1～6.5MPPS となっておりますが、DLL 内部ではモーションデバイス LSI ： G9003 の「FL/FH/FA 速度レジスタ」と「速度倍率レジスタ」にデータを計算し設定しております。　それらのレジスタ値と出力パルス速度は以下の関係となっております。

「出力パルス速度」 = 「FL/FH/FA 速度レジスタ設定値」×「速度倍率」

例を挙げると速度倍率が 10 倍、出力パルス速度が 200,000PPS である場合

200,000(出力パルス速度) = 20,000(速度レジスタ設定値) × 10 倍(速度倍率)

となります。

この倍率という考え方により速度レジスタのビット長の制限を越えた速度設定を行えるようになっています。具体的には各速度設定レジスタ設定範囲は 1～100,000 であり、それに 0.1～66.6 の倍率を乗ずることで 0.1～6.5MPPS という速度設定範囲を実現しています。

各速度倍率とその出力パルス速度の範囲を下表に示します。

倍率	出力速度範囲(PPS)	倍率	出力速度範囲(PPS)
0.1	0.1 ～ 10,000.0	5	5 ～ 500,000
0.2	0.2 ～ 20,000.0	10	10 ～ 1,000,000
0.5	0.5 ～ 50,000.0	20	20 ～ 2,000,000
1	1 ～ 100,000	50	50 ～ 5,000,000
2	2 ～ 200,000	66.6	66.6‥～ 6,666,666.6‥

倍率はパルス出力中に可変するものではなく、動作スタート前に設定された値に固定されます。

それぞれの倍率は表の通り範囲を持ち、一つの動作の中では範囲を超えることはできません。

本 DLL では MNETM1SetSpeed 関数で速度データが設定されるたび、速度データ構造体メンバー「SpdL」,「SpdH」,「SpdA」に設定されたデータの中で、もっと大きい値を速度倍率選択速度とし、その速度倍率選択速度を満たす最も小さい速度倍率が自動的に選択され、各速度レジスタの値が算出/設定されています。(小さい速度倍率のほうが設定できる速度の速度間隔が細かくなるため)

基本的には自動的に選択される速度倍率は意識する必要はありませんが､以下の場合に注意が必要です。

①速度のオーバーライドを行う場合、速度倍率を任意に指定(固定)したい場合。

前述の通り各速度倍率にはそれぞれ範囲があります。起動時に選択される速度倍率の範囲を超えてオーバーライドを行いたい場合は､MNETM1FixSpeedRange 関数を使用し対象デバイスの速度倍率を指定する値に固定することができます。あらかじめオーバーライド速度を含む速度倍率に固定することで倍率自動設定ではできない速度オーバーライドが行えます。

MNETM1FixSpeedRange 関数による速度倍率の固定は MNETM1UnFixSpeedRange 関数で固定設定が解除されるまで有効となります。また既に固定設定の状態で改めて MNETM1FixSpeedRange 関数を使用することで固定する速度倍率を変更できます。

②設定速度データの丸めについて。特に小さい速度を設定する場合

各速度レジスタへの設定値(整数値)は「各 SpdL/SpdH/SpdA 設定値」÷「速度倍率」となります。

したがって SpdL/SpdH/SpdA に設定した値は丸められます（小数部切捨て)。

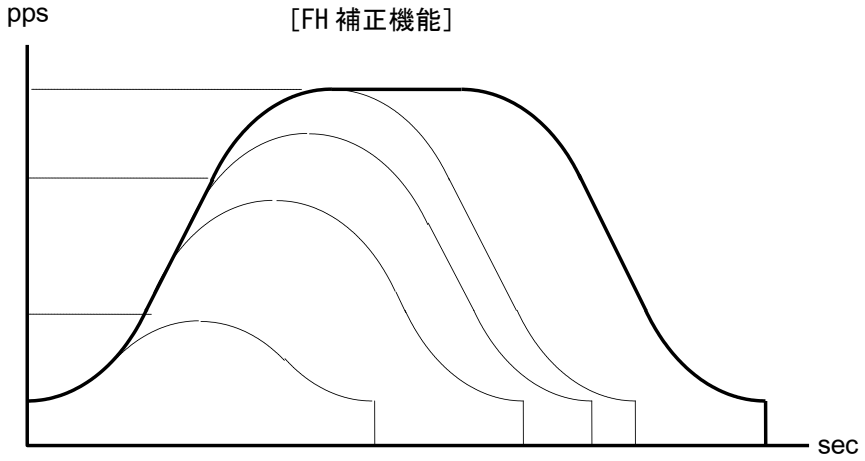
DLL 内部パラメータ「速度設定自動補正」が"補正しない"設定で SpdL/SpdH/SpdA 設定した値が 0 ではないのに、設定速度が 0 に丸められた場合、速度計算結果エラーとなります。小さい速度を設定する場合､前述の通り設定する 3 つの速度の中で最も大きい速度が､速度倍率選択速度となりますので､速度の組み合わせにはご注意ください。

(DLL 内部パラメータ「速度設定自動補正」が"補正する"設定で SpdL/SpdH/SpdA 設定した値が 0 ではないのに丸められた値が 0 の場合、設定速度は選択された倍率の最小値に補正されます。)

(例) SpdH：4,000,000PPS, SpdL: 30PPS の場合、速度倍率は 50 倍が選択されます。このとき、FL の設定値は 30÷50 で 0 となり，速度計算結果エラーとなります。

6－2－10－2. FH補正

FH 補正とは MNETM1Inc 関数、MNETM1Abs 関数での位置決め動作で加減速動作時に移動量が少なく、設定した FH 速度に到達できず三角駆動となる場合、自動的に FH 速度を低下させて三角駆動を回避する機能です。



移動量による最高速度の自動修正

FH 補正機能の有効/無効は MNETM1SetPar 関数の項目「FH 補正有効/無効」で設定します。

識別子	値	内容	設定範囲	設定値／取得値 （網掛け：デフォルト）
MNETM1_PAR_FH_CORRECT_DIS	306h	FH補正有効/無効	0/1	0:有効, 1:無効

注意

S 字動作で減速時間が加速時間の 2 倍以上(減速レートが加速レートの 1/2 以下)の場合、つまり減速が長い加減速設定の場合、FH 補正有効/無効にかかわらず、FH 補正が行われます。

6−2−11. SD入力

SD信号の入力によりFH速度で動作中の軸をFL速度に減速することができます。
(SD信号入力による減速を行う場合はあらかじめMNETM1SetSpeed関数でスタートモードに「加減速」を指定しFL速度および加減速データの設定を行っておく必要があります。スタートモード「FL定速」,「FH定速」で速度データ設定を行った場合、加減速データは設定されません。)
(6−2−3. 速度データ構造体)-「設定上の注意」-「③動作起動関数の引数「スタートモード」との関係」を参照ください。
SD信号入力の有効/無効、信号論理、他のパラメータはMNETM1SetPar関数で設定します。

SD入力処理に関連するパラメータは以下の通りです。

識別子	値	内容	設定範囲	設定値／取得値（網掛け：デフォルト）
MNETM1_PAR_SD_ENA	310h	SD有効	0/1	0:無効, 1:有効
MNETM1_PAR_SDLTC_ENA	311h	SDラッチ有効(注)	0/1	0:無効, 1:有効
MNETM1_PAR_SD_LOGIC	312h	SD信号論理	0/1	0:負論理, 1:正論理
MNETM1_PAR_SD_PROC	313h	SD入力時処理	0/1	0:減速のみ, 1:減速停止

(注)SDラッチが無効である場合、SD入力中のみFL速度までスローダウンし、SD入力がOFFするとFH速度まで再加速します。
SDラッチが有効である場合、動作停止までSD入力はラッチされSD入力がOFFしても再加速しません。

6−2−12. PCS入力

PCS信号の入力によりPCS信号がONした位置を基準にした位置決めができます。
PCS信号入力の有効/無効、信号論理、他のパラメータはMNETM1SetPar関数で設定します。

PCS入力処理に関連するパラメータは以下の通りです。

識別子	値	内容	設定範囲	設定値／取得値（網掛け：デフォルト）
MNETM1_PAR_PCS_ENA	31Dh	PCS信号有効	0/1	0:無効, 1:有効
MNETM1_PAR_PCS_LOGIC	31Eh	PCS信号論理	0/1	0:負論理, 1:正論理

PCS動作を行う手順

(MNETM1SetPar関数でPCS信号を「有効」に設定します。)
① MNETM1Inc関数にPCS入力ONからの位置決め量をセットし実行します。
② 指定した速度で軸が動作し、PCS入力がONになった地点から位置決めが行われます。
(MNETM1SetPar関数でPCS信号を「無効」に設定します。)

またMNETM1SoftPCS関数を使用することで、PCS入力を代行するコマンドを発行できます。

6−2−13. 同時スタート

同時スタート動作とは、複数の軸を同時スタート(CSTA)信号待ち状態にし、外部からの同時スタート(CSTA)信号の入力により、一斉にスタートさせる動作です。

CSTA 信号入力の有効/無効、信号論理のパラメータは MNETM1SetPar 関数で設定します。

CSTA 入力処理に関連するパラメータは以下の通りです。

識別子	値	内容	設定範囲	設定値／取得値 (網掛け：デフォルト)
MNETM1_PAR_CSTA_ENA	380h	CSTA有効	0/1	0:無効, 1:有効
MNETM1_PAR_CSTA_TYPE	381h	CSTA入力仕様	0/1	0:レベルトリガ 1:エッジトリガ

同時スタート動作を行う手順

(MNETM1SetPar 関数で CSTA を「有効」に設定します。)

① CSTA設定が「有効」の軸は移動命令を実行すると、同時スタート(CSTA)信号入力待ち状態となります。(軸の状態はMNETM1GetBitDataStruct関数などで取得可能です)

② 同時スタート(CSTA)信号の入力により、動作が開始されます。

(MNETM1SetPar 関数で CSTA を「無効」に設定します。)

同時スタート待ち状態の解除

同時スタート待ち状態の解除はMNETM1Stop関数を停止モード：「即停止」で実行することで行われます。

6−2−14. 同時停止

同時停止は、同時停止(CSTP)信号の入力により複数軸を同時停止させる機能です。

CSTP信号の有効無効および、信号論理などのパラメータはMNETM1SetPar関数で設定します。

CSTP 入力処理に関連するパラメータは以下の通りです。

識別子	値	内容	設定範囲	設定値／取得値 (網掛け：デフォルト)
MNETM1_PAR_CSTP_ENA	382h	CSTP有効	0/1	0:無効, 1:有効
MNETM1_PAR_AUTO_CSTA_ENA	383h	CSTP自動出力有効	0/1	0:無効, 1:有効 (異常停止時にCSTP自動出力)
MNETM1_PAR_CSTP_MODE	384h	CSTP入力時処理	0/1	0:即停止, 1:減速停止

同時ストップ動作を行う手順

(MNETM1SetPar 関数で CSTP を「有効」に設定します。)

CSTP設定が「有効」の軸は、動作中に同時ストップ(CSTP)信号入力がONした場合、設定した入力時処理で停止します。

(MNETM1SetPar 関数で CSTA を「無効」に設定します。)

6－2－15. 基本固定パラメータ

本 DLL でモーションデバイスを操作する場合に、設定が必要な基本固定パラメータの設定内容は以下の通りです。モーションデバイス電源リセット、および MNETM1Reset 関数などでのリセット後、モーションデバイス複合関数ご利用前に必ずこの基本固定パラメータを設定ください。

識別子	値	内容	設定範囲	設定値
MNETM1_PAR_BASE_DATA	0h	基本固定パラメータ設定	0	0を設定ください

設定される内容

①軸動作スタート時 SEND 信号をリセットする
②出力パルス Duty50%
③汎用入出力設定 P0-P3:出力、P4-P7:入力
④CLR 入力でカウンタ 1-3 リセット
⑤原点復帰完了でカウンタ 1-3 リセット
（カウンタリセットタイミングについては(6－2－6. 原点復帰モード)の項を参照ください)
⑥補正移動時もカウンタ 1-3 は動作を行う
⑦±EL,SD,ORG,ALM,INP,EMG 入力にフィルタを挿入します。

6－2－16. コンパレータ

各モーションデバイスは 28 ビットのコンパレータを 3 つ内蔵しており、カウンタ値との比較を行うことができます。　各コンパレータは比較カウンタとしてそれぞれカウンタ 1～3 を選択できます。
また、色々な比較方法と 3 種類の条件成立時の処理方法が用意されております。

コンパレータの利用により以下の動作(処理)を行うことができます。

◆イベント割込を発生、比較結果の外部出力(CMP1-3)
◆動作の即停止／減速停止
◆コンパレータ 1, 2 を利用した、ソフトリミット機能
◆カウンタ 3(偏差)とコンパレータを利用した、ステッピングモータの脱調検出
◆カウンタ 3(汎用)とコンパレータ 3 を利用した、IDX(同期) 信号出力

これらコンパレータの動作仕様は MNETM1SetPar 関数で設定します。

識別子	値	内容	設定範囲	設定値／取得値 （網掛け：デフォルト）
MNETM1_PAR_CMP1_CNT	360h	コンパレータ1比較カウンタ	0～3	0:カウンタ1 1:カウンタ2 2:カウンタ3 3:常に条件不成立
MNETM1_PAR_CMP1_MODE	361h	コンパレータ1比較方法	0～7	1:CMP1データ=比較カウンタ （カウント方向無関係） 2:CMP1データ=比較カウンタ （カウントアップ中） 3:CMP1データ=比較カウンタ （カウントダウン中） 4:CMP1データ>比較カウンタ 5:CMP1データ<比較カウンタ 6:+ソフトエンドリミットに設定 （CMP1 < カウンタ1） その他:常に不成立
MNETM1_PAR_CMP1_PROC	362h	コンパレータ1成立処理	0～2	0:処理無し(INT,CP出力は有) 1:即停止 2:減速停止
MNETM1_PAR_CMP1_DATA	363h	コンパレータ1データ	右記	-134,217,728～134,217,727 （8000000h～7ffffffh） 0
MNETM1_PAR_CMP2_CNT	364h	コンパレータ2比較カウンタ	0～3	0:カウンタ1 1:カウンタ2 2:カウンタ3 3:常に条件不成立
MNETM1_PAR_CMP2_MODE	365h	コンパレータ2比較方法	0～7	1:CMP2データ=比較カウンタ （カウント方向無関係） 2:CMP2データ=比較カウンタ （カウントアップ中） 3:CMP2データ=比較カウンタ （カウントダウン中） 4:CMP2データ>比較カウンタ 5:CMP2データ<比較カウンタ 6:-ソフトエンドリミットに設定 （CMP2 > カウンタ1） その他:常に不成立
MNETM1_PAR_CMP2_PROC	366h	コンパレータ2成立処理	0～2	0:処理無し(INT,CP出力は有) 1:即停止 2:減速停止
MNETM1_PAR_CMP2_DATA	367h	コンパレータ2データ	右記	-134,217,728～134,217,727 （8000000h～7ffffffh） 0
MNETM1_PAR_CMP3_CNT	368h	コンパレータ3比較カウンタ	0～3	0:カウンタ1 1:カウンタ2 2:カウンタ3 3:常に条件不成立

識別子	値	内容	設定範囲	設定値／取得値 （網掛け：デフォルト）
MNETM1_PAR_CMP3_MODE	369h	コンパレータ3比較方法	0～15	1:CMP3データ=比較カウンタ （カウント方向無関係） 2:CMP3データ=比較カウンタ （カウントアップ中） 3:CMP3データ=比較カウンタ （カウントダウン中） 4:CMP3データ>比較カウンタ 5:CMP3データ<比較カウンタ 7:設定禁止 8:IDX(同期)信号出力 （カウント方向無関係） 9:IDX(同期)信号出力 （カウントアップ中） 10:IDX(同期)信号出力 （カウントダウン中） その他:常に不成立
MNETM1_PAR_CMP3_PROC	36Ah	コンパレータ3成立処理	0～2	0:処理無し(INT,CP出力は有) 1:即停止 2:減速停止
MNETM1_PAR_CMP3_DATA	36Bh	コンパレータ3データ	右記	-134,217,728～134,217,727 (8000000h～7ffffffh) 0

Motionnet®
Remote I/O & Remote Motion

6−2−17. ソフトリミット

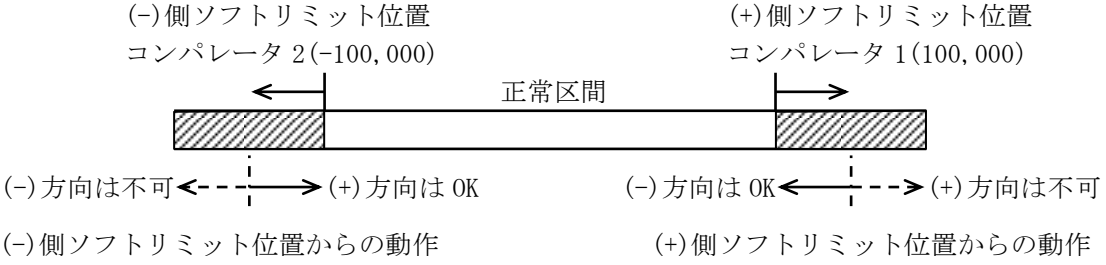
コンパレータ 1 と 2 の利用によりソフトリミット機能を設定できます。
ソフトリミット動作時には下記の処理を行う事ができます。
①パルス出力を即停止させる。
②パルス出力を減速停止させる。
また、スタートコマンド書き込み時にソフトリミットが ON 状態の場合、ソフトリミットが ON 状態の方向へはスタートは出来ません。逆方向へはスタートできます。

以下のデータの設定はすべて MNETM1SetPar 関数で行います。

ソフトリミット機能を有効にする手順
①コンパレータ値の設定
(コンパレータ 1 に+ソフトリミットとする値を、コンパレータ 2 に-ソフトリミットとする値を設定します。)
②コンパレータ比較カウンタの設定
(コンパレータ 1, コンパレータ 2 共に「カウンタ 1」を比較カウンタに設定します。)
③コンパレータ成立時処理の設定
(ソフトエンドリミット到達時(コンパレータ 1, コンパレータ 2 成立時)に行う処理(即停止/減速停止)を設定します。)
④コンパレータ比較方法の設定
(コンパレータ 1, コンパレータ 2 の比較方法に「ソフトエンドリミット」(設定値:6)を設定します。)

ソフトリミット機能を無効にする手順
コンパレータ比較方法の設定
(コンパレータ 1, コンパレータ 2 の比較方法に「常に不成立」(設定値:0)を設定します。)

設定例
コンパレータ 1 設定値 = 100,000 :(+)側リミット値
コンパレータ 2 設定値 = -100,000 :(−)側リミット値



ソフトリミット機能利用上の注意
ソフトリミットを有効にしている場合は原点復帰動作中も機能します。
機械条件的に単一のソフトリミットで対応できない場合、原点復帰動作中はソフトリミットを無効にしたり、原点復帰動作中は別のソフトリミット値を設定するなどして対応ください。

6−2−18. イベント割り込み設定

イベント割り込み要因の設定は MNETM1SetPar 関数で行います。
イベント割り込みを発生させたい内容に対応するビットを１にしたデータを設定ください。

Bit	ﾋﾞｯﾄ名	内容
0	IREN	正常停止時
1	IRUS	加速開始時
2	IRUE	加速終了時
3	IRDS	減速開始時
4	IRDE	減速終了時
5	IRC1	コンパレータ 1 条件成立時
6	IRC2	コンパレータ 2 条件成立時
7	IRC3	コンパレータ 3 条件成立時
8	IRCL	CLR 信号入力によるカウント値のリセット時
9	IRLT	LTC 入力によるカウント値のラッチ時
10	IROL	ORG 入力によるカウント値のラッチ時
11	IRSD	SD 入力 ON 時
12	IRSA	CSTA 入力 ON 時
31-13	未定義	（常に 0 を設定して下さい。）

イベント割り込み要因の読出しは MNETM1GetRIST 関数で行います。

6−2−19. ラッチ

全てのカウンタは下記 5 通りの方法でカウント値をラッチでき、ラッチした値の読出しは MNETM1GetLTC 関数で行います。

①LTC 信号 OFF→ON 時
②ORG 信号 OFF→ON 時
③コンパレータ 2 条件成立時
④コンパレータ 3 条件成立時
⑤MNETM1SoftLTC 関数の実行(ラッチコマンドの書込み)

ラッチ動作の設定/処理などは MNETM1SetPar 関数で設定します。

ラッチに関連するパラメータは以下の通りです。

識別子	値	内容	設定範囲	設定値／取得値 (網掛け：デフォルト)
MNETM1_PAR_LTC_LOGIC	31Fh	LTC信号論理	0/1	0:立下り, 1:立上り
MNETM1_PAR_LTC_MODE	320h	LTCタイミング	0～3	0:LTC入力時 1:ORG入力時 2:コンパレータ2条件成立時 3:コンパレータ3条件成立時
MNETM1_PAR_LTC_SPD	321h	速度ラッチ	0/1	0:無効, 1:有効
MNETM1_PAR_LTC_DIS	322h	ハードウェアによる ラッチ(上記①-④) 無効	0/1	0:有効, 1:無効

6－2－20. 速度オーバーライド

加減速動作の場合、MNETM1OvrideSpeed 関数で動作中の速度変更を行うことができます。

ただし、位置決め動作(MNETM1Inc, MNETM1Abs)で、残りパルスが少ない場合、あるいは減速中での速度変更はエラーとなり受け付けられません。

速度変更時の加減速度は動作スタート時に設定した加減速設定となります。

(速度設定時の加減速指定方法が「時間」であっても、その際に計算された加減速比率が適用されます)

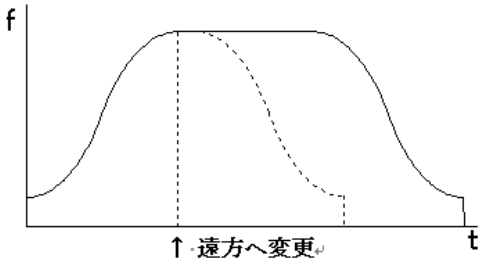
オーバーライドできる速度は動作スタート時の速度倍率によります。

速度オーバーライドを行う場合は(6－2－10－1. 速度倍率)の項を必ず参照ください。

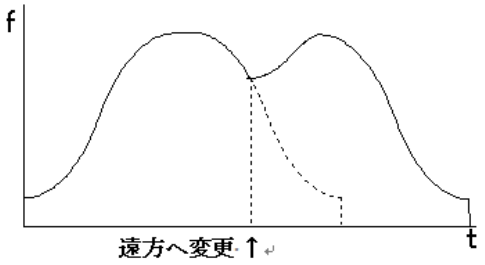
6－2－21. 目標位置オーバーライド

MNETM1OvridePos 関数を使用することで、目標位置のオーバーライドができます。

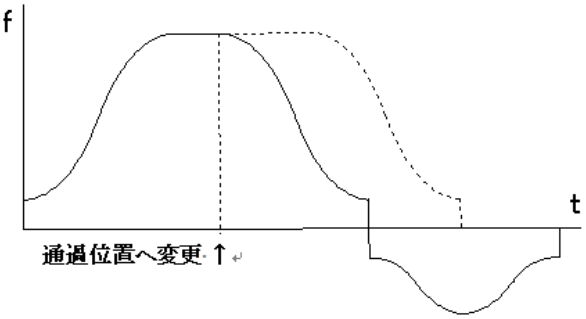
加速・定速動作中に新データを最初の目標位置より遠方に変更した場合、そのままの速度パターンで動作し新データ位置で位置決め完了になります。



減速中に新データを最初の目標位置より遠方に変更した場合、その位置から FH 速度に再加速後、新データ位置で位置決め完了になります。



新データ位置をすでに通過している場合、または減速中で最初の目標位置より手前に変更した場合、減速停止後に逆転して新データ位置で位置決め完了になります。



加速／減速動作を行うのは高速スタート時のみです。また、目標位置のオーバーライドは、位置決め動作完了まで何度でも可能です。

位置のオーバーライドは、動作中(FL/FH 定速動作中,加/減速中,バックラッシュ補正動作)にだけ有効です。

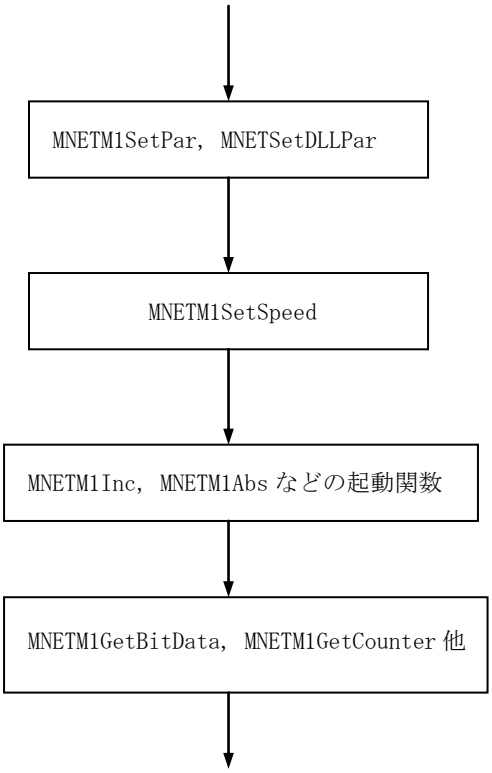
停止直前にオーバーライドを行った場合、オーバーライドが受け付けられない場合があります。

停止直前にオーバーライドを行う可能性がある場合、オーバーライドの成否判定は停止時の位置により判断して下さい。あるいはオーバーライドに失敗した場合はエラー割り込みが発生し、エラー割り込み要因レジスタの対応するビットが ON します。したがって MNETM1GetREST 関数でのエラー割り込み要因の読出しでも行うことができます。

6－2－22. 操作の流れ

システム起動から 1 軸モーションデバイスを操作する流れの概略は次のようになります。

ボードのオープン，システム通信，I/O 通信開始



ローカルデバイスのパラメータデータを設定します。
(信号仕様、信号論理、動作仕様などの環境設定)

速度データの設定。
(速度、加減速度、S 字動作設定など)

動作の起動。
(移動量、動作方向などを設定し動作を起動します)

モーションデバイスの状態取得。
動作状態，信号状態，カウンタなどのモニタ

6−2−22−1. プログラム概略

Motionnet の 1 軸モーションデバイスを操作するプログラムコードの概略を示します。
以下のコードはあくまで流れをつかんでもらうためのものであり、関数の実行結果の取得などをはじめとする実際の動作において必要な処理の多くを省略してありますのでご注意ください。

[VisualC++]

```
short lineNo=0;            // ライン番号
short devNo=0;             // デバイス番号
TMNETM1Cmd cmd = {0};      // 速度データ構造体
TMNETM1BitData bit = {0}; // ビットデータ構造体
bool posEnd = false;       // 位置決め完了状態

PPCI_L112Open();                    // ボードのオープン
MNETCenterReset(lineNo);            // センターデバイスリセット
MNETSysComm(lineNo);                // システム通信
MNETIOCommStart(lineNo);            // I/O 通信スタート

// パラメータ設定(下記はその一部です。サンプルプログラムを参照ください)
MNETM1SetPar(lineNo, devNo, MNETM1_PAR_BASEDATA, 0);          // 共通基本データ
MNETM1SetPar(lineNo, devNo, MNETM1_PAR_OUTPLS_TYPE, 4);       // 出力パルス仕様(設定:4)
MNETM1SetPar(lineNo, devNo, MNETM1_PAR_ORG_LOGIC, 0);         // ORG 論理(負論理)
MNETM1SetPar(lineNo, devNo, MNETM1_PAR_ALM_LOGIC, 1);         // ALM 論理(正論理)
MNETM1SetPar(lineNo, devNo, MNETM1_PAR_ALM_PROC, 0);          // ALM 入力時処理(即停止)
MNETM1SetPar(lineNo, devNo, MNETM1_PAR_INP_ENA, 0);           // INP 有効(無効)
MNETM1SetPar(lineNo, devNo, MNETM1_PAR_INP_LOGIC, 0);         // INP 論理(負論理)
MNETM1SetPar(lineNo, devNo, MNETM1_PAR_EL_PROC, 0);           // EL 入力時処理(即停止)
MNETM1SetPar(lineNo, devNo, MNETM1_PAR_ORG_MODE, 0);          // 原点復帰方法(設定:0)
MNETM1SetPar(lineNo, devNo, MNETM1_PAR_EZ_LOGIC, 0);          // Z 相論理(立下り)

// 速度データ設定
cmd.SpdL = 500;                     // FL 速度:500PPS
cmd.SpdH = 3000;                    // FH 速度:3000PPS
cmd.AdcMtd = MNET_ADC_BY_TIME;      // 加減速データ指定方法:時間
cmd.Acc = 0.5;                      // 加速時間:0.5Sec
cmd.Dec = 0.5;                      // 減速時間:0.5Sec
cmd.SMode = MNET_NON_S;             // S 字未使用
cmd.StartMode = MNET_STAMODE_UD;    // スタートモード:加減速
MNETM1SetSpeed(lineNo, devNo, &cmd);              // 速度データ設定

MNETM1Inc(lineNo, devNo, MNET_STAMODE_UD, 20000);  // INC 位置決め移動起動

// 位置決め完了を待つ
while (posEnd == false) {
  MNETM1GetBitDataStruct(lineNo, devNo, &bit);     // ビットデータ読出し
  if (bit.send) posEnd= true;
}

MNETCenterReset(lineNo);            // センターデバイスリセット
PPCI_L112Close();                   // ボードのクローズ
```

[VisualBasic]

```
  Dim lineNo As Integer      ’ライン番号
  Dim devNo As Integer       ’デバイス番号
  Dim cmd As TMNETM1Cmd      ’速度データ構造体
  Dim bit As TMNETM1BitData  ’ビットデータ構造体
  Dim posEnd As Boolean      ’位置決め完了状態

  PPCI_L112Open                    ’ボードのオープン
  lineNo = 0
  devNo = 0
  MNETCenterReset lineNo           ’センターデバイスリセット
  MNETSysComm lineNo               ’システム通信
  MNETIOCommStart lineNo           ’I/O 通信スタート

  ’パラメータ設定(下記はその一部です。サンプルプログラムを参照ください)
  MNETM1SetPar lineNo, devNo, MNETM1_PAR_BASEDATA, 0          ’共通基本データ
  MNETM1SetPar lineNo, devNo, MNETM1_PAR_OUTPLS_TYPE, 4       ’力パルス仕様(設定:4)
  MNETM1SetPar lineNo, devNo, MNETM1_PAR_ORG_LOGIC, 0         ’ORG 論理(負論理)
  MNETM1SetPar lineNo, devNo, MNETM1_PAR_ALM_LOGIC, 1         ’ALM 論理(正論理)
  MNETM1SetPar lineNo, devNo, MNETM1_PAR_ALM_PROC, 0          ’ALM 入力時処理(即停止)
  MNETM1SetPar lineNo, devNo, MNETM1_PAR_INP_ENA, 0           ’INP 有効(無効)
  MNETM1SetPar lineNo, devNo, MNETM1_PAR_INP_LOGIC, 0         ’INP 論理(負論理)
  MNETM1SetPar lineNo, devNo, MNETM1_PAR_EL_PROC, 0           ’EL 入力時処理(即停止)
  MNETM1SetPar lineNo, devNo, MNETM1_PAR_ORG_MODE, 0          ’原点復帰方法(設定:0)
  MNETM1SetPar lineNo, devNo, MNETM1_PAR_EZ_LOGIC, 0          ’Z 相論理(立下り)

  ’速度データ設定
  cmd.SpdL = 500                   ’FL 速度:500PPS
  cmd.SpdH = 3000                  ’FH 速度:3000PPS
  cmd.AdcMtd = MNET_ADC_BY_TIME    ’加減速データ指定方法:時間
  cmd.Acc = 0.5                    ’加速時間:0.5Sec
  cmd.Dec = 0.5                    ’減速時間:0.5Sec
  cmd.SMode = MNET_NON_S           ’S 字未使用
  cmd.StartMode = MNET_STAMODE_UD  ’スタートモード:加減速
  MNETM1SetSpeed lineNo, devNo, cmd                ’速度データ設定

  MNETM1Inc lineNo, devNo, MNET_STAMODE_UD, 20000   ’INC 位置決め移動起動

  ’位置決め完了を待つ
  Do While (posEnd = False)
    MNETM1GetBitDataStruct lineNo, devNo, bit      ’ビットデータ読出し
  If (bit.send) Then
      posEnd = True
    EndIf
  Loop

  MNETCenterReset 0                ’センターデバイスリセット
  PPCI_L112Close                   ’ボードのクローズ
```

[VB.NET]

```
    Dim lineNo As Short         ’ライン番号
    Dim devNo As Short          ’デバイス番号
    Dim cmd As TMNETM1Cmd       ’速度データ構造体
    Dim bit As TMNETM1BitData   ’ビットデータ構造体
    Dim posEnd As Boolean       ’位置決め完了状態

    PPCI_L112Open()                     ’ボードのオープン
    lineNo = 0
    devNo = 0
    MNETCenterReset(lineNo)             ’センターデバイスリセット
    MNETSysComm(lineNo)                 ’システム通信
    MNETIOCommStart(lineNo)             ’I/O 通信スタート

    ’パラメータ設定(下記はその一部です。サンプルプログラムを参照ください)
    MNETM1SetPar(lineNo, devNo, MNETM1_PAR_BASEDATA, 0)          ’共通基本データ
    MNETM1SetPar(lineNo, devNo, MNETM1_PAR_OUTPLS_TYPE, 4)       ’出力パルス仕様(設定:4)
    MNETM1SetPar(lineNo, devNo, MNETM1_PAR_ORG_LOGIC, 0)         ’ORG 論理(負論理)
    MNETM1SetPar(lineNo, devNo, MNETM1_PAR_ALM_LOGIC, 1)         ’ALM 論理(正論理)
    MNETM1SetPar(lineNo, devNo, MNETM1_PAR_ALM_PROC, 0)          ’ALM 入力時処理(即停止)
    MNETM1SetPar(lineNo, devNo, MNETM1_PAR_INP_ENA, 0)           ’INP 有効(無効)
    MNETM1SetPar(lineNo, devNo, MNETM1_PAR_INP_LOGIC, 0)         ’INP 論理(負論理)
    MNETM1SetPar(lineNo, devNo, MNETM1_PAR_EL_PROC, 0)           ’EL 入力時処理(即停止)
    MNETM1SetPar(lineNo, devNo, MNETM1_PAR_ORG_MODE, 0)          ’原点復帰方法(設定:0)
    MNETM1SetPar(lineNo, devNo, MNETM1_PAR_EZ_LOGIC, 0)          ’Z 相論理(立下り)

    ’速度データ設定
    cmd.SpdL = 500                      ’FL 速度:500PPS
    cmd.SpdH = 3000                     ’FH 速度:3000PPS
    cmd.AdcMtd = MNET_ADC_BY_TIME       ’加減速データ指定方法:時間
    cmd.Acc = 0.5                       ’加速時間:0.5Sec
    cmd.Dec = 0.5                       ’減速時間:0.5Sec
    cmd.SMode = MNET_NON_S              ’S 字未使用
    cmd.StartMode = MNET_STAMODE_UD     ’スタートモード:加減速
    MNETM1SetSpeed(lineNo, devNo, cmd)                  ’速度データ設定

    MNETM1Inc(lineNo, devNo, MNET_STAMODE_UD, 20000)    ’INC 位置決め移動起動

    ’位置決め完了を待つ
    Do While (posEnd = False)
      MNETM1GetBitDataStruct(lineNo, devNo, bit)        ’ビットデータ読出し
    If (bit.send) Then
        posEnd = True
      EndIf
    Loop

    MNETCenterReset(0)                  ’センターデバイスリセット
    PPCI_L112Close()                    ’ボードのクローズ
```

6−3. データ定義

6−3−1. ビットデータ構造体

ビットデータの読出しに使用します。(MNETM1GetBitDataStruct 関数使用時)
ビットデータを一括で取得することが可能です。

書式

[VisualC++]

```
//1軸モーションボードビットデータ取得用構造体
struct TMNETM1BitData {
  short mpout1;  //MPOUT1(SVON)
  short mpout2;  //MPOUT2(ALMRES)
  short out1;    //OUT1
  short out2;    //OUT2
  short in1;     //IN1
  short in2;     //IN2
  short mpin;    //MPIN(SVRDY)
  short in3;     //IN3

  short sint;    //SINT(割り込み発生)
  short send;    //SEND(動作停止)
  short serr;    //SERR(エラー割り込み発生)
  short sevt;    //SEVT(イベント割り込み発生)
  short sbsy;    //SBSY(BUSY)

  short cnd;     //軸動作状態
  short sdir;    //動作方向
  short salm;    //ALM入力
  short spel;    //PEL(プラスエンドリミット)入力
  short smel;    //MEL(マイナスエンドリミット)入力
  short sorg;    //ORG入力
  short sdin;    //SD入力
  short ssta;    //STA入力
  short sstp;    //STP入力
  short semg;    //EMG入力
  short spcs;    //PCS入力
  short serc;    //ERC出力
  short sez;     //EZ入力
  short sclr;    //CLR入力
  short sltc;    //LTC入力
  short sinp;    //INP入力
  short scp1;    //コンパレータ1比較条件成立
  short scp2;    //コンパレータ2比較条件成立
  short scp3;    //コンパレータ3比較条件成立
};
```

[VisualBasic]

```
  Type TMNETM1BitData
    mpout1 As Integer  '/MPOUT1(SVON)
    mpout2 As Integer  '/MPOUT2(ALMRES)
    out1 As Integer    '/OUT1
    out2 As Integer    '/OUT2
    in1 As Integer     '/IN1
    in2 As Integer     '/IN2
    mpin As Integer    '/MPIN(SVRDY)
    in3 As Integer     '/IN3

    sint As Integer    '/SINT(割り込み発生)
    send As Integer    '/SEND(動作停止)
    serr As Integer    '/SERR(エラー割り込み発生)
    sevt As Integer    '/SEVT(イベント割り込み発生)
    sbsy As Integer    '/SBSY(BUSY)

    cnd As Integer     '/軸動作状態
    sdir As Integer    '/動作方向
    salm As Integer    '/ALM 入力
    spel As Integer    '/PEL(プラスエンドリミット)入力
    smel As Integer    '/MEL(マイナスエンドリミット)入力
    sorg As Integer    '/ORG 入力
    sdin As Integer    '/SD 入力
    ssta As Integer    '/STA 入力
    sstp As Integer    '/STP 入力
    semg As Integer    '/EMG 入力
    spcs As Integer    '/PCS 入力
    serc As Integer    '/ERC 出力
    sez As Integer     '/EZ 入力
    sclr As Integer    '/CLR 入力
    sltc As Integer    '/LTC 入力
    sinp As Integer    '/INP 入力
    scp1 As Integer    '/コンパレータ 1 比較条件成立
    scp2 As Integer    '/コンパレータ 2 比較条件成立
    scp3 As Integer    '/コンパレータ 3 比較条件成立
  End Type
```

[VB.NET]

```
Structure TMNETM1BitData
  Dim Mpout1 As Short '/ MPOUT1(SVON)
  Dim Mpout2 As Short '/ MPOUT2(ALMRES)
  Dim out1 As Short '/ OUT1
  Dim out2 As Short '/ OUT2
  Dim in1 As Short '/ IN1
  Dim in2 As Short '/ IN2
  Dim mpin As Short '/ MPIN(SVRDY)
  Dim in3 As Short '/ IN3
  Dim Sint As Short '/ SINT(割り込み発生)
  Dim Send As Short '/ SEND(動作停止)
  Dim Serr As Short '/ SERR(エラー割り込み発生)
  Dim sevt As Short '/ SEVT(イベント割込み発生)
  Dim sbsy As Short '/ SBSY(BUSY)
  Dim Cnd As Short '/ 軸動作状態
  Dim Sdir As Short '/ 動作方向
  Dim Salm As Short '/ ALM 入力
  Dim Spel As Short '/ PEL(プラスエンドリミット)入力
  Dim Smel As Short '/ MEL(マイナスエンドリミット)入力
  Dim Sorg As Short '/ ORG 入力
  Dim Sdin As Short '/ SD 入力
  Dim Ssta As Short '/ STA 入力
  Dim Sstp As Short '/ STP 入力
  Dim Semg As Short '/ EMG 入力
  Dim Spcs As Short '/ PCS 入力
  Dim Serc As Short '/ ERC 出力
  Dim Sez As Short '/ EZ 入力
  Dim Sclr As Short '/ CLR 入力
  Dim Sltc As Short '/ LTC 入力
  Dim Sinp As Short '/ INP 入力
  Dim Scp1 As Short '/ コンパレータ 1 比較条件成立
  Dim Scp2 As Short '/ コンパレータ 2 比較条件成立
  Dim Scp3 As Short '/ コンパレータ 3 比較条件成立
End Structure
```

[メンバー]

メンバー	内容	取得値
mpout1	mpout1 信号状態	0:OFF, 1:ON
mpout2	mpout2 信号状態	0:OFF, 1:ON
out1	out1 信号状態	0:OFF, 1:ON
out2	out2 信号状態	0:OFF, 1:ON
in1	in1 信号状態	0:OFF, 1:ON
in2	in2 信号状態	0:OFF, 1:ON
mpin	mpin 信号状態	0:OFF, 1:ON
in3	in3 信号状態	0:OFF, 1:ON
sint	sint ビット状態(割込有)	0:OFF, 1:ON
send	send ビット状態(動作停止)	0:OFF, 1:ON
serr	serr ビット状態(エラー割込有)	0:OFF, 1:ON
sevt	sevt ビット状態(イベント割込有)	0:OFF, 1:ON
sbsy	sbsy ビット状態(動作中)	0:OFF, 1:ON
cnd	軸動作状態	下表参照
sdir	動作方向	0:+方向, 1:-方向
salm	ALM 信号状態	0:OFF, 1:ON
spel	+EL 信号状態	0:OFF, 1:ON
smel	-EL 信号状態	0:OFF, 1:ON
sorg	ORG 信号状態	0:OFF, 1:ON
sdin	SD 信号状態	0:OFF, 1:ON
ssta	STA 信号状態	0:OFF, 1:ON
sstp	STP 信号状態	0:OFF, 1:ON
semg	EMG 信号状態	0:OFF, 1:ON
spcs	PCS 信号状態	0:OFF, 1:ON
serc	ERC 信号状態	0:OFF, 1:ON
sez	EZ 信号状態	0:OFF, 1:ON
sclr	CLR 信号状態	0:OFF, 1:ON
sltc	LTC 信号状態	0:OFF, 1:ON
sinp	INP 信号状態	0:OFF, 1:ON
scp1	CMP1 比較条件成立状態	0:非成立, 1:成立
scp2	CMP2 比較条件成立状態	0:非成立, 1:成立
scp3	CMP3 比較条件成立状態	0:非成立, 1:成立

信号状態は論理パラメータが加味された結果となります。
またこの値はLSIとしての値となります。ボードとしての信号論理は各I/F回路を御確認ください。

cnd(軸動作状態)一覧

取得値	状態	取得値	状態
0	停止中	8	加速中
1	STA 入力待ち	9	FH 定速動作中
2	ERC タイマ完了待ち	10	減速中
3	方向変化タイマ完了待ち	11	INP 入力待ち
4	バックラッシュ補正中	12	その他(スタート制御中)
5	PA/PB 入力待ち	13	未定義
6	FA 定速動作中	14	未定義
7	FL 定速動作中	15	未定義

6−3−2. ステータス／ビットマップ

6−3−2−1. メインステータス

モーションデバイスLSI:G9003のメインステータスはI/Oポートの下位2ポートに割り付けられており、MNETGetLocalData関数でデータを取得できます。

Bit	ﾋﾞｯﾄ名	内容
0	SINT	割り込み発生により1になります。(ビット1,2,3のOR(いずれかが1でON))
1	SEND	動作停止により割り込みを発生し、1になります。割り込みリセットコマンド(MNETResetSEND関数実行)で0になります。
2	SERR	エラー割り込み発生により1になり、RESTレジスタ読み出し(MNETM1GetREST関数実行)で0になります。
3	SEVT	イベント割り込み発生により1になり、RISTレジスタ読み出し(MNETM1GetRIST関数実行)で0になります。
7～4	未定義	常に0になります。
8	SBSY	パルス出力開始で1になります。動作停止で0になります。 (=BSY)
15～9	未定義	常に0になります。

6−3−2−2. エラー割り込み要因

エラー割り込み要因のマップです。

エラー割り込みが発生したときに対応するビットが1になります。

データの取得はMNETM1GetREST関数で行います。

このデータはモーションデバイスLSI:G9003のRESTレジスタ(エラー割り込みステータス)から取得しています。 このレジスタは読出しによりリセットされます。

Bit	ﾋﾞｯﾄ名	内容
0	ESC1	コンパレータ1条件成立による停止時 (+SL)
1	ESC2	コンパレータ2条件成立による停止時 (-SL)
2	ESC3	コンパレータ3条件成立による停止時
3	ESPL	＋EL入力ONによる停止時
4	ESML	－EL入力ONによる停止時
5	ESAL	ALM入力ONによる停止時
6	ESSP	STP入力ONによる停止時
7	ESEM	EMG入力ONによる停止時
8	ESSD	SD入力ONによる減速停止時
9	ESPO	PA/PB入力用バッファカウンタのオーバーフロー発生時
10	ESNT	通信エラー発生よる停止時
11	未定義	(常に0になります。)
12	ESOR	位置のオーバーライドが実行できなかった時
13	ESEE	EA/EB入力エラー発生時
14	ESPE	PA/PB入力エラー発生時
31～15	未定義	未定義 (常に0になります。)

6−3−2−3. イベント割り込み要因

イベント割り込み要因のマップです。

MNETM1SetPar 関数で設定したイベントが発生したときに対応するビットが 1 になります。

データの取得は MNETM1GetRIST 関数で行います。

このデータはモーションデバイス LSI:G9003 の RIST レジスタ(イベント割り込みステータス)から取得しています。このレジスタは読出しによりリセットされます。

Bit	ﾋﾞｯﾄ名	内容
0	ISEN	自動停止時
1	ISUS	加速開始時
2	ISUE	加速終了時
3	ISDS	減速開始時
4	ISDE	減速終了時
5	ISC1	コンパレータ 1 条件成立時
6	ISC2	コンパレータ 2 条件成立時
7	ISC3	コンパレータ 3 条件成立時
8	ISCL	CLR 信号入力によるカウント値のリセット時
9	ISLT	LTC 入力によるカウント値のラッチ時
10	ISOL	ORG 入力によるカウント値のラッチ時
11	ISSD	SD 入力 ON 時
12	ISSA	STA 入力 ON 時
31～13	未定義	未定義（常に 0 になります。）

6−3−3. 識別子

6−3−3−1. スタートモード

スタートモードは下記 3 パターンがあります。

MNETM1SetSpeed 関数での速度データ設定時に速度データ構造体のメンバー「StartMode」に、および移動命令関数の引数に設定するデータとその識別子は次のようになります。

識別子	値	内容
MNET_STAMODE_FL	0050h	FL定速
MNET_STAMODE_FH	0051h	FH 定速
MNET_STAMODE_UD	0053h	加減速

6−3−3−2. 動作方向

連続動作や原点復帰動作など、動作方向を指定する移動命令の場合に、速度データ構造体のメンバー「Dir」に設定します。

識別子	値	内容
MNET_DIR_PLUS	0	プラス方向
MNET_DIR_MINUS	1	マイナス方向

6−3−3−3. 加減速指定方法

MNETM1SetSpeed 関数などでの加減速データ設定時、加減速度を加速(減速)時間で指定するか、加速(減速)レート(1 秒あたりの増分(減分)パルス)で指定するかを、速度データ構造体のメンバー「AdcMtd」に設定します。

識別子	値	内容
MNET_ADC_BY_RATE	0	レート指定(1秒あたりの増分(減分)パルス速度)
MNET_ADC_BY_TIME	1	時間指定　(秒)

6−3−3−4. S 字使用/未使用

MNETM1SetSpeed 関数などでの速度データ設定時、S 字動作を使用するかどうかを、速度データ構造体のメンバー「SMode」に設定します。

識別子	値	内容
MNET_NON_S	0	S字動作なし
MNET_USE_S	1	S字動作あり

6−3−3−5. カウンタ指定

モーションデバイスのカウンタデータ取得/設定時に、対象カウンタを指定します。

識別子	値	内容
MNET_SEL_COUNTER1	0	カウンタ1選択
MNET_SEL_COUNTER2	1	カウンタ2選択
MNET_SEL_COUNTER3	2	カウンタ3選択

6−3−3−6. ラッチ指定

モーションデバイスのラッチデータ取得時に、対象のラッチを指定します。

識別子	値	内容
MNET_SEL_LTC1	0	ラッチ1選択
MNET_SEL_LTC2	1	ラッチ2選択
MNET_SEL_LTC3	2	ラッチ3選択

6−3−3−7. 停止モード

MNETM1Stop 関数使用時に停止方法を指定します。

識別子	値	内容
MNET_STOP	0	即停止
MNET_SDSTOP	1	減速停止
MNET_EMGSTOP	2	非常停止

6−3−3−8. 速度倍率レンジ

MNETM1FixSpeedRange 関数使用時に速度倍率を指定します。

識別子	値	内容
MNETM1_SPEED_01_RANGE	0	速度倍率:0.1倍
MNETM1_SPEED_02_RANGE	1	速度倍率:0.2倍
MNETM1_SPEED_05_RANGE	2	速度倍率:0.5倍
MNETM1_SPEED_1_RANGE	3	速度倍率:1倍
MNETM1_SPEED_2_RANGE	4	速度倍率:2倍
MNETM1_SPEED_5_RANGE	5	速度倍率:5倍
MNETM1_SPEED_10_RANGE	6	速度倍率:10倍
MNETM1_SPEED_20_RANGE	7	速度倍率:20倍
MNETM1_SPEED_50_RANGE	8	速度倍率:50倍
MNETM1_SPEED_66_RANGE	9	速度倍率:66.6倍

6−3−3−9. ビット選択

MNETM1GetBitData, MNETM1SetBitData 関数使用時に対象ビットを指定します。

識別子	値	内容	取得(設定値)
MNETM1_SEL_BIT_MPOUT1	0	MPOUT1 信号状態	0:OFF, 1:ON **(Set可能)**
MNETM1_SEL_BIT_MPOUT2	1	MPOUT2 信号状態	0:OFF, 1:ON **(Set可能)**
MNETM1_SEL_BIT_OUT1	2	OUT1 信号状態	0:OFF, 1:ON **(Set可能)**
MNETM1_SEL_BIT_OUT2	3	OUT2 信号状態	0:OFF, 1:ON **(Set可能)**
MNETM1_SEL_BIT_IN1	4	IN1 信号状態	0:OFF, 1:ON
MNETM1_SEL_BIT_IN2	5	IN2 信号状態	0:OFF, 1:ON
MNETM1_SEL_BIT_MPIN	6	MPIN 信号状態	0:OFF, 1:ON
MNETM1_SEL_BIT_IN3	7	IN3 信号状態	0:OFF, 1:ON
MNETM1_SEL_BIT_SINT	10	SINT ビット状態(割込有)	0:OFF, 1:ON
MNETM1_SEL_BIT_SEND	11	SEND ビット状態(動作停止)	0:OFF, 1:ON
MNETM1_SEL_BIT_SERR	12	SERR ビット状態(エラー割込有)	0:OFF, 1:ON
MNETM1_SEL_BIT_SEVT	13	SEVT ビット状態(イベント割込有)	0:OFF, 1:ON
MNETM1_SEL_BIT_SBSY	14	SBSY ビット状態(動作中)	0:OFF, 1:ON
MNETM1_SEL_BIT_CND	20	CND(軸動作状態)	6-3-1.ビットデータ構造体参照
MNETM1_SEL_BIT_SDIR	21	動作方向	0:+方向, 1:-方向
MNETM1_SEL_BIT_SALM	22	ALM 信号状態	0:OFF, 1:ON
MNETM1_SEL_BIT_SPEL	23	+EL 信号状態	0:OFF, 1:ON
MNETM1_SEL_BIT_SMEL	24	-EL 信号状態	0:OFF, 1:ON
MNETM1_SEL_BIT_SORG	25	ORG 信号状態	0:OFF, 1:ON
MNETM1_SEL_BIT_SDIN	26	SD 信号状態	0:OFF, 1:ON
MNETM1_SEL_BIT_SSTA	27	CSTA 信号状態	0:OFF, 1:ON
MNETM1_SEL_BIT_SSTP	28	CSTP 信号状態	0:OFF, 1:ON
MNETM1_SEL_BIT_SEMG	29	EMG 信号状態	0:OFF, 1:ON
MNETM1_SEL_BIT_SPCS	30	PCS 信号状態	0:OFF, 1:ON
MNETM1_SEL_BIT_SERC	31	ERC 信号状態	0:OFF, 1:ON **(Set可能)**
MNETM1_SEL_BIT_SEZ	32	EZ 信号状態	0:OFF, 1:ON
MNETM1_SEL_BIT_SCLR	33	CLR 信号状態	0:OFF, 1:ON
MNETM1_SEL_BIT_SLTC	34	LTC 信号状態	0:OFF, 1:ON
MNETM1_SEL_BIT_SINP	35	INP 信号状態	0:OFF, 1:ON
MNETM1_SEL_BIT_SCP1	36	CMP1 比較条件成立状態	0:非成立, 1:成立
MNETM1_SEL_BIT_SCP2	37	CMP2 比較条件成立状態	0:非成立, 1:成立
MNETM1_SEL_BIT_SCP3	38	CMP3 比較条件成立状態	0:非成立, 1:成立

信号状態は論理パラメータが加味された結果となります。
またこの値はLSIとしての値となります。ユニットとしての信号論理は各I/F回路を御確認ください。

6−3−3−10. パラメータ項目番号定義

MNETM1GetPar 関数, MNETM1SetPar 関数使用時に対象のパラメータ項目を指定します。

識別子	値	内容	設定範囲	設定値／取得値（網掛け：デフォルト(リセット時)）
MNETM1_PAR_BASE_DATA	0h	基本固定パラメータ(注1)	0	0を設定ください
MNETM1_PAR_OUTPLS_TYPE	301h	出力パルス仕様	0～7	当項下部記載の出力パルス仕様参照 0
MNETM1_PAR_DIR_TIM_ENA	302h	方向変化タイマ有効	0/1	0:無効, 1:有効
MNETM1_PAR_IDLPLS	303h	アイドリングパルス	0～7	アイドリングパルス数 0
MNETM1_PAR_VIBE_SUPPRESS1	304h	振動抑制1	右記	当項下部記載の振動抑制参照 0
MNETM1_PAR_VIBE_SUPPRESS2	305h	振動抑制2	右記	当項下部記載の振動抑制参照 0
MNETM1_PAR_FH_CORRECT_DIS	306h	FH補正有効/無効	0/1	0:有効, 1:無効
MNETM1_PAR_SD_ENA	310h	SD有効	0/1	0:無効, 1:有効
MNETM1_PAR_SDLTC_ENA	311h	SDラッチ有効	0/1	0:無効, 1:有効
MNETM1_PAR_SD_LOGIC	312h	SD信号論理	0/1	0:負論理, 1:正論理
MNETM1_PAR_SD_PROC	313h	SD入力時処理	0/1	0:減速のみ, 1:減速停止
MNETM1_PAR_ORG_LOGIC	314h	ORG信号論理	0/1	0:負論理, 1:正論理
MNETM1_PAR_ALM_LOGIC	315h	ALM信号論理	0/1	0:負論理, 1:正論理
MNETM1_PAR_ALM_PROC	316h	ALM入力時処理	0/1	0:即停止, 1:減速停止
MNETM1_PAR_ERC_LOGIC	317h	ERC信号論理	0/1	0:負論理, 1:正論理
MNETM1_PAR_ERCPLS_LEN	318h	ERCパルス幅	0～7	0:12μs, 1:102μs, 2:409μs 3:1.6ms 4:13ms, 5:52ms, 6:104ms, 7:ﾚﾍﾞﾙ出力
MNETM1_PAR_ERR_ERC_PROC	319h	エラー停止時ERC出力	0/1	0:しない, 1:出力する
MNETM1_PAR_ORG_ERC_PROC	31Ah	原点復帰時ERC出力	0/1	0:しない, 1: 出力する
MNETM1_PAR_INP_ENA	31Bh	INP信号有効	0/1	0:無効, 1:有効
MNETM1_PAR_INP_LOGIC	31Ch	INP信号論理	0/1	0:負論理, 1:正論理
MNETM1_PAR_PCS_ENA	31Dh	PCS信号有効	0/1	0:無効, 1:有効
MNETM1_PAR_PCS_LOGIC	31Eh	PCS信号論理	0/1	0:負論理, 1:正論理
MNETM1_PAR_LTC_LOGIC	31Fh	LTC信号論理	0/1	0:立下り, 1:立上り
MNETM1_PAR_LTC_MODE	320h	LTCタイミング	0～3	0:LTC入力時 , 1:ORG入力時 2:コンパレータ2条件成立時 3:コンパレータ3条件成立時
MNETM1_PAR_LTC_SPD	321h	速度ラッチ	0/1	0:無効, 1:有効
MNETM1_PAR_LTC_DIS	322h	ハードウェアによるラッチ無効	0/1	0:有効, 1:無効
MNETM1_PAR_EL_PROC	323h	EL入力時処理	0/1	0:即停止, 1:減速停止 (注2,3)

注１．本 DLL の仕様により決まっている固定パラメータ設定です。 本 DLL 使用時には必ず設定ください。

注 2. EL 入力 ON 時の処理を減速停止に設定した場合、EL 入力 ON で減速を開始しますので、EL 位置を通過して停止します。機械系への衝突等に十分御注意下さい。

注 3. 停止方法を減速停止に設定した場合には、減速して停止するまで入力 ON 状態を保持して下さい。
正常停止と異常停止との判別は停止タイミングの後で決定されます。そのため、加減速位置決め動作で減速中に異常停止信号が入力されても正常停止と認識する場合があります。
その場合でも異常停止信号を入力し続けていれば、次動作は継続せずに異常停止になります。

識別子	値	内容	設定範囲	設定値／取得値 （網掛け：デフォルト （リセット時））
MNETM1_PAR_INP_FILT_ENA	324h	入力フィルタ	0/1	0:有効, 1:無効
MNETM1_PAR_ENC_TYPE	330h	エンコーダ仕様	0～3	0:90度位相差1逓倍 1:90度位相差2逓倍 2:90度位相差4逓倍 3:EAの立上りでカウントアップ, EBの立上りでカウントダウン
MNETM1_PAR_ENC_REV	331h	エンコーダ入力反転	0/1	0:無効, 1:有効
MNETM1_PAR_ENC_FILT_ENA	332h	エンコーダフィルタ	0/1	0:有効, 1:無効
MNETM1_PAR_PULSER_TYPE	340h	パルサー仕様	0～3	0:90度位相差1逓倍 1:90度位相差2逓倍 2:90度位相差4逓倍 3:PAの立上りでカウントアップ, PBの立上りでカウントダウン
MNETM1_PAR_PULSER_REV	341h	パルサー入力反転	0/1	0:無効, 1:有効
MNETM1_PAR_PULSER_DIV	342h	パルサー分周比	0～2047	設定値/2048に分周されます 0設定時は分周されません。0
MNETM1_PAR_PULSER_MULT	343h	パルサー逓倍比	0～31	(設定値+1)倍に逓倍されます。0
MNETM1_PAR_PULSER_FILT_ENA	344h	パルサーフィルタ	0/1	0:有効, 1:無効
MNETM1_PAR_PULSER_CNT3_MODE	345h	カウンタ3入力選択	0～6	0:出力パルス 1:EA/EB入力 2:PA/PB入力 3:クロックの1/4096分周クロック 4:出力パルスとEA/EBの偏差 5:出力パルスとPA/PBの偏差 6:EA/EBとPA/PBの偏差
MNETM1_PAR_ORG_MODE	350h	原点復帰方法	0～12	当項下部記載の原点復帰方法参照 0
MNETM1_PAR_EZ_CNT	351h	Z相回数	0～15	0(1回)～15(16回)となります。0
MNETM1_PAR_EZ_LOGIC	352h	Z相論理	0/1	0:立下り, 1:立上り
MNETM1_PAR_CMP1_CNT	360h	コンパレータ1比較カウンタ	0～3	0:カウンタ1 1:カウンタ2 2:カウンタ3 3:常に条件不成立
MNETM1_PAR_CMP1_MODE	361h	コンパレータ1比較方法	0～7	1:CMP1データ=比較カウンタ (カウント方向無関係) 2:CMP1データ=比較カウンタ (カウントアップ中) 3:CMP1データ=比較カウンタ (カウントダウン中) 4:CMP1データ>比較カウンタ 5:CMP1データ<比較カウンタ 6:+ソフトエンドリミットに設定 (CMP1 < カウンタ1) その他:常に不成立

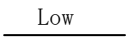
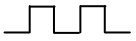
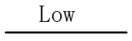
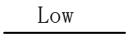
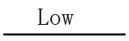
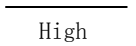
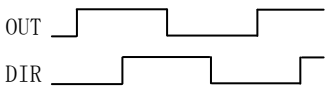
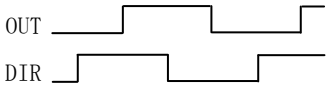
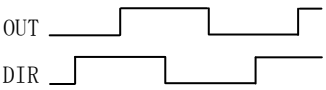
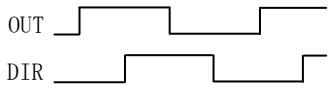
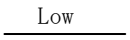
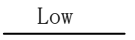
Motionnet®
Remote I/O & Remote Motion

識別子	値	内容	設定範囲	設定値／取得値 （網掛け：デフォルト （リセット時））
MNETM1_PAR_CMP1_PROC	362h	コンパレータ1成立処理	0～2	0:処理無し(INT,CP出力は有) 1:即停止 2:減速停止
MNETM1_PAR_CMP1_DATA	363h	コンパレータ1データ	右記	-134,217,728～134,217,727 (8000000h～7ffffffh) 0
MNETM1_PAR_CMP2_CNT	364h	コンパレータ2比較カウンタ	0～3	0:カウンタ1 1:カウンタ2 2:カウンタ3 3:常に条件不成立
MNETM1_PAR_CMP2_MODE	365h	コンパレータ2比較方法	0～7	1:CMP2データ=比較カウンタ （カウント方向無関係） 2:CMP2データ=比較カウンタ （カウントアップ中） 3:CMP2データ=比較カウンタ （カウントダウン中） 4:CMP2データ>比較カウンタ 5:CMP2データ<比較カウンタ 6:-ソフトエンドリミットに設定 （CMP2 > カウンタ1） その他:常に不成立
MNETM1_PAR_CMP2_PROC	366h	コンパレータ2成立処理	0～2	0:処理無し(INT,CP出力は有) 1:即停止 2:減速停止
MNETM1_PAR_CMP2_DATA	367h	コンパレータ2データ	右記	-134,217,728～134,217,727 (8000000h～7ffffffh) 0
MNETM1_PAR_CMP3_CNT	368h	コンパレータ3比較カウンタ	0～3	0:カウンタ1 1:カウンタ2 2:カウンタ3 3:常に条件不成立
MNETM1_PAR_CMP3_MODE	369h	コンパレータ3比較方法	0～15	1:CMP3データ=比較カウンタ （カウント方向無関係） 2:CMP3データ=比較カウンタ （カウントアップ中） 3:CMP3データ=比較カウンタ （カウントダウン中） 4:CMP3データ>比較カウンタ 5:CMP3データ<比較カウンタ 7:設定禁止 8:IDX(同期)信号出力 （カウント方向無関係） 9:IDX(同期)信号出力 （カウントアップ中） 10:IDX(同期)信号出力 （カウントダウン中） その他:常に不成立
MNETM1_PAR_CMP3_PROC	36Ah	コンパレータ3成立処理	0～2	0:処理無し(INT,CP出力は有) 1:即停止 2:減速停止
MNETM1_PAR_SET_CSTA_ENA	380h	CSTA有効	0/1	0:無効, 1:有効

識別子	値	内容	設定範囲	設定値／取得値 （網掛け：デフォルト （リセット時））
MNETM1_PAR_SET_CSTA_TYPE	381h	CSTA入力仕様	0/1	0:レベルトリガ, 1:エッジトリガ
MNETM1_PAR_SET_CSTP_ENA	382h	CSTP有効	0/1	0:無効, 1:有効
MNETM1_PAR_SET_AUTO_ CSTA_ENA	383h	CSTP自動出力有効	0/1	0:無効, 1:有効 （異常停止時にCSTP自動出力）
MNETM1_PAR_SET_CSTP_MODE	384h	CSTP入力時処理	0/1	0:即停止, 1:減速停止

出力パルスタイプ

出力パルス仕様を設定します。

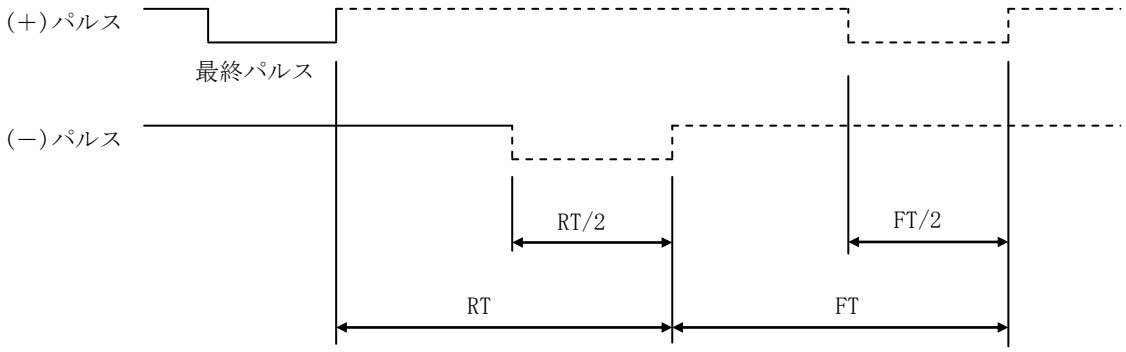
設定/取得値	（＋）方向動作時		（－）方向動作時	
	PULS 信号	DIR 信号	PULS 信号	DIR 信号
0				
1				
2				
3				
4				
5				
6				
7				

Motionnet®
Remote I/O & Remote Motion

振動抑制

項目	設定内容
振動抑制1	下図の RT の時間を設定します。設定単位は内部基準クロック周期の 64 倍(CLK=40Mbps で約 1.6μs です) RT 設定範囲:0～65,535 -> 設定時間範囲:0～約 0.1sec となります。
振動抑制2	下図の FT の時間を設定します。設定単位は内部基準クロック周期の 64 倍(CLK=40Mbps で約 1.6μs です) FT 設定範囲:0～65,535 -> 設定時間範囲:0～約 0.1sec となります。

下図の波線部分が振動抑制機能により付加されるパルスです。(＋方向動作の例)



6−3−3−11. DLL 内部パラメータ項目番号定義

MNETM1SetDLLPar 関数使用時に対象のパラメータ項目を指定します。

識別子	値	内容	設定範囲	設定値／取得値(網掛け：デフォルト)
MNETM1_DLLPAR_PREMOVE_CHK	3E0h	軸動作前確認省略	0/1	0:省略しない,1:省略
MNETM1_DLLPAR_ADC_CORRECT	3E1h	加減速設定自動補正	0/1	0:補正する,1:補正しない
MNETM1_DLLPAR_SPD_CORRECT	3E2h	速度設定自動補正	0/1	0:補正する,1:補正しない

Motionnet® Remote I/O & Remote Motion

6−3−4. 戻り値

識別子	値	内容
MNET_ERR_DEV_NOT_MOTION	3000h	デバイスがモーションデバイス(G9003)でない
MNET_ERR_M1_SET_STAMODE	3001h	スタートモード指定エラー
MNET_ERR_M1_SET_SMODE	3002h	S字モード指定エラー
MNET_ERR_M1_SET_DIR	3003h	方向指定エラー
MNET_ERR_M1_SET_ADCMTD	3004h	加減速指定方法指定エラー
MNET_ERR_M1_SET_SPD_DATA	3005h	速度指定エラー
MNET_ERR_M1_SET_ADC_DATA	3006h	加減速度指定エラー
MNET_ERR_M1_SET_SSECT	3007h	S字範囲指定エラー
MNET_ERR_M1_CALC_ACC_DATA	3008h	加速データ計算エラー(設定値が大きすぎ/小さすぎ)
MNET_ERR_M1_CALC_DEC_DATA	3009h	減速データ計算エラー(設定値が大きすぎ/小さすぎ)
MNET_ERR_M1_CALC_SPD_DATA	300Ah	速度計算結果エラー
MNET_ERR_M1_STATUS_NOT_READY	3020h	受付状態にない(動作中など)
MNET_ERR_M1_STATUS_ALM	3021h	受付状態にない(ALM)
MNET_ERR_M1_STATUS_EL	3022h	受付状態にない(EL)
MNET_ERR_M1_STATUS_EMG	3023h	受付状態にない(EMG)
MNET_ERR_M1_STATUS_CSTP	3024h	受付状態にない(CSTP)
MNET_ERR_M1_STATUS_ALREADY	3025h	すでに目的の状態にある
MNET_ERR_M1_POSDATA_SOFTLIMIT	3026h	指令値ソフトエンドリミット
MNET_ERR_M1_FAILED_OVRPOS	3026h	位置オーバーライド失敗
MNET_ERR_M1_FAILED_OVRSPD	3027h	速度オーバーライド失敗
MNET_ERR_M1_FAILED_FILEOPEN	3030h	ファイルオープン失敗
MNET_ERR_M1_INVALID_FILEDATA	3031h	ファイルデータ不正

7. ローカル 4 軸(6143)モーションボード機能

7−1. 関数詳細

関数詳細説明にあたって

(1) 軸パターンについて

軸パターンとは、4 軸を下記のように 4 ビットのデータで表し、複数軸を同時に対象として選択するデータです。

[例]

軸 0 と軸 2 の場合 ： 5h

軸3	軸2	軸1	軸0
0	1	0	1

軸 0 の場合 ： 1h

軸3	軸2	軸1	軸0
0	0	0	1

軸 0 と軸 2 と軸 3 の場合 ： Dh

軸3	軸2	軸1	軸0
1	1	0	1

7－1－1. ローカル 4 軸(6143)モーションボード関数

7－1－1－1. MNETM4SetSpeed

機能

速度データ設定

書式

[VisualC++]

```
short __stdcall MNETM4SetSpeed(
      short lineNo,
      short devNo,
      short axPtn,
      TMNETM4Spd* spd,
      short staMode);
```

[VisualBasic]

```
Declare Function MNETM4SetSpeed Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axPtn As Integer,
      ByRef spd As TMNETM4Spd,
      ByVal staMode As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM4SetSpeed Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axPtn As Short,
      ByRef spd As TMNETM4Spd,
      ByVal staMode As Short) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

axPtn ***軸パターン***

spd ***速度データ構造体*** (7－2－3. 速度データ構造体参照)

staMode ***スタートモード***

識別子	値	内容
MNET_STAMODE_FL	0050h	FL定速
MNET_STAMODE_FH	0051h	FH 定速
MNET_STAMODE_UD	0053h	加減速

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

例① スタートモード ： FL 定速

[VisualC++]

```
TMNETM4Spd spd = {0};              // 速度データ構造体定義, 初期化(0リセット)

spd.SpdL[0] = 100;                 // 軸0 FL速度   :   100PPS
spd.SpdL[1] = 200;                 // 軸1 FL速度   :   200PPS
MNETM4SetSpeed(0, 3, 0x3, &spd, MNET_STAMODE_FL);
```

[VisualBasic]

```
Dim spd As TMNETM4Spd              ’速度データ構造体定義, 初期化(0リセット)

spd.SpdL(0) = 100                  ’軸0 FL速度   :   100PPS
spd.SpdL(1) = 200                  ’軸1 FL速度   :   200PPS
MNETM4SetSpeed 0, 3, &H3, spd, MNET_STAMODE_FL
```

[VB.NET]

```
Dim spd As New TMNETM4Spd          ’速度データ構造体定義, 初期化(0リセット)
spd.Initialize()                   ’構造体初期化
spd.SpdL(0) = 100                  ’軸0 FL速度   :   100PPS
spd.SpdL(1) = 200                  ’軸1 FL速度   :   200PPS
MNETM4SetSpeed(0, 3, &H3, spd, MNET_STAMODE_FL)
```

スタートモード:FL 定速でライン 0,デバイス 3 の軸 0,1 の速度を設定します。

例② スタートモード : 加減速

[VisualC++]

```
TMNETM4Spd spd = {0};                  // 速度データ構造体定義, 初期化(0リセット)

spd.SpdL[0] = 100;                     // 軸0 FL速度   :   100PPS
spd.SpdH[0] = 3000;                    // 軸0 FH速度   :   3000PPS
spd.AdcMtd[0] = MNET_ADC_BY_TIME;      // 軸0 加減速指定方法  :  時間指定
spd.Smode[0] = MNET_NON_S;             // 軸0 S字設定  :   S字なし
spd.Acc[0] = 0.5;                      // 軸0 加速時間 :   0.5秒
spd.Dec[0] = 0.5;                      // 軸0 減速時間 :   0.5秒

spd.SpdL[2] = 2000;                    // 軸2 FL速度   :   2000PPS
spd.SpdH[2] = 10000;                   // 軸2 FH速度   :   10000PPS
spd.AdcMtd[2] = MNET_ADC_BY_RATE;      // 軸2 加減速指定方法  :  レート指定
spd.Smode[2] = MNET_USE_S;             // 軸2 S字設定   :   S字
spd.Acc[2] = 1000;                     // 軸2 加速レート:   1000PPS/秒
spd.Dec[2] = 5000;                     // 軸2 減速レート:   5000PPS/秒

MNETM4SetSpeed(0, 3, 0x5, &spd, MNET_STAMODE_UD);
```

[VisualBasic]

```
Dim spd As TMNETM4Spd                  ’速度データ構造体定義, 初期化(0リセット)

spd.SpdL(0) = 100                      ’軸0 FL速度   :   100PPS
spd.SpdH(0) = 3000                     ’軸0 FH速度   :   3000PPS
spd.AdcMtd(0) = MNET_ADC_BY_TIME       ’軸0 加減速指定方法  :  時間指定
spd.Smode(0) = MNET_NON_S              ’軸0 S字設定  :   S字なし
spd.Acc(0) = 0.5                       ’軸0 加速時間 :   0.5秒
spd.Dec(0) = 0.5                       ’軸0 減速時間 :   0.5秒

spd.SpdL(2) = 2000                     ’軸2 FL速度   :   2000PPS
spd.SpdH(2) = 10000                    ’軸2 FH速度   :   10000PPS
spd.AdcMtd(2) = MNET_ADC_BY_RATE       ’軸2 加減速指定方法  :  レート指定
spd.Smode(2) = MNET_USE_S              ’軸2 S字設定   :   S字
spd.Acc(2) = 1000                      ’軸2 加速レート:   1000PPS/秒
spd.Dec(2) = 5000                      ’軸2 減速レート:   5000PPS/秒

MNETM4SetSpeed 0, 3, &H5, spd, MNET_STAMODE_UD
```

[VB.NET]

```
Dim spd As New TMNETM4Spd              ’速度データ構造体定義, 初期化(0リセット)
spd.Initialize()                       ’構造体初期化

spd.SpdL(0) = 100                      ’軸0 FL速度   :   100PPS
spd.SpdH(0) = 3000                     ’軸0 FH速度   :   3000PPS
spd.AdcMtd(0) = MNET_ADC_BY_TIME       ’軸0 加減速指定方法  :  時間指定
spd.Smode(0) = MNET_NON_S              ’軸0 S字設定  :   S字なし
spd.Acc(0) = 0.5                       ’軸0 加速時間 :   0.5秒
spd.Dec(0) = 0.5                       ’軸0 減速時間 :   0.5秒

spd.SpdL(2) = 2000                     ’軸2 FL速度   :   2000PPS
spd.SpdH(2) = 10000                    ’軸2 FH速度   :   10000PPS
spd.AdcMtd(2) = MNET_ADC_BY_RATE       ’軸2 加減速指定方法  :  レート指定
spd.Smode(2) = MNET_USE_S              ’軸2 S字設定  :   S字
spd.Acc(2) = 1000                      ’軸2 加速レート:   1000PPS/秒
spd.Dec(2) = 5000                      ’軸2 減速レート:   5000PPS/秒

MNETM4SetSpeed(0, 3, &H5, spd, MNET_STAMODE_UD)
```

スタートモード:加減速でライン 0,デバイス 3 の軸 0,2 の速度を設定します。

7－1－1－2. MNETM4PtpInc

機能

PtpInc移動(補間なし相対位置移動)を起動します。

書式

[VisualC++]

```
short __stdcall MNETM4PtpInc(
      short lineNo,
      short devNo,
      short axPtn,
      TMNETM4Dst dst);
```

[VisualBasic]

```
Declare Function MNETM4PtpInc Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axPtn As Integer,
      ByRef dst As TMNETM4Dst) As Integer
```

[VB.NET]

```
Declare Function MNETM4PtpInc Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axPtn As Short,
      ByRef dst As TMNETM4Dst) As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axPtn	***軸パターン***
dst	***移動データ構造体***（7－2－4. 移動データ構造体参照）

備考

速度データはMNETM4SetSpeed関数で設定ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照）

使用例

例① スタートモード ： FL定速

[VisualC++]

```
TMNETM4Dst dst = {0};              // 移動データ構造体定義, 初期化(0リセット)

dst.Pos[0] = 1000;                 // 軸0移動量 ： 1000パルス
dst.Pos[1] = -5000;                // 軸1移動量 ： -5000パルス
dst.StartMode = MNET_STAMODE_FL;   // スタートモード設定
MNETM4PtpInc(0, 3, 0x3, &dst);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst              ’移動データ構造体定義, 初期化(0リセット)

dst.Pos(0) = 1000                  ’軸0移動量 ： 1000パルス
dst.Pos(1) = -5000                 ’軸1移動量 ： -5000パルス
dst.StartMode = MNET_STAMODE_FL    ’スタートモード設定
MNETM4PtpInc 0, 3, &H3, dst
```

[VB.NET]

```
Dim dst As New TMNETM4Dst          ’移動データ構造体定義, 初期化(0リセット)
dst.Initialize()                   ’構造体初期化

dst.Pos(0) = 1000                  ’軸0移動量 ： 1000パルス
dst.Pos(1) = -5000                 ’軸1移動量 ： -5000パルス
dst.StartMode = MNET_STAMODE_FL    ’スタートモード設定
MNETM4PtpInc(0, 3, &H3, dst)
```

ライン 0,デバイス 3, 軸 0,1 にスタートモード:FL 定速で PtpInc 移動指令を発行します。

Motionnet®
Remote I/O & Remote Motion

例② スタートモード ： 加減速

[VisualC++]

```
TMNETM4Dst dst = {0};              // 移動データ構造体定義, 初期化(0リセット)

dst.Pos[2] = -1000;                // 軸2移動量 : -1000パルス
dst.Pos[3] = -5000;                // 軸3移動量 : -5000パルス
dst.StartMode = MNET_STAMODE_UD;   // スタートモード設定
MNETM4PtpInc(0, 3, 0xC, &dst);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst              ’移動データ構造体定義, 初期化(0リセット)

dst.Pos(2) = -1000                 ’軸2移動量 : -1000パルス
dst.Pos(3) = -5000                 ’軸3移動量 : -5000パルス
dst.StartMode = MNET_STAMODE_ UD   ’スタートモード設定
MNETM4PtpInc 0, 3, &HC, dst
```

[VB.NET]

```
Dim dst As New TMNETM4Dst          ’移動データ構造体定義, 初期化(0リセット)
dst.Initialize()                   ’構造体初期化

dst.Pos(2) = -1000                 ’軸2移動量 : -1000パルス
dst.Pos(3) = -5000                 ’軸3移動量 : -5000パルス
dst.StartMode = MNET_STAMODE_ UD   ’スタートモード設定
MNETM4PtpInc(0, 3, &HC, dst)
```

ライン 0,デバイス 3, 軸 2,3 にスタートモード:加減速で PtpInc 移動指令を発行します。

Motionnet®
Remote I/O & Remote Motion

7－1－1－3. MNETM4LineInc

機能

直線補間Inc移動(相対位置指定 補間移動)を起動します。

書式

[VisualC++]

```
short __stdcall MNETM4LineInc(
        short lineNo,
        short devNo,
        short axPtn,
        TMNETM4Dst dst
        TMNETM4Spd spd);
```

[VisualBasic]

```
Declare Function MNETM4LineInc Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal axPtn As Integer,
        ByRef dst As TMNETM4Dst,
        ByRef spd As TMNETM4Spd) As Integer
```

[VB.NET]

```
Declare Function MNETM4LineInc Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal axPtn As Short,
        ByRef dst As TMNETM4Dst,
        ByRef spd As TMNETM4Spd) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン***
dst	***移動データ構造体*** (7－2－4. 移動データ構造体参照)
spd	***速度データ構造体*** (7－2－3. 速度データ構造体参照)

備考

・動作速度は軸パターンに関わらず、速度データ構造体の軸0のデータに設定します。
・指定する動作速度は全補間軸間のベクトル速度(合成速度)となります。
・直線補間動作関数の実行によりMNETM4SetSpeed関数で設定した、非補間動作用の速度データは未設定状態となります。
・直線補間時は、補間に含まれる軸の以下の設定が強制的に有効となります。
　＊ 同時停止(CSTP)信号による停止。
　＊ 異常停止時、同時停止(CSTP)信号自動出力
　＊ 同時減速(CSD)信号による減速
　＊ 減速時、FL定速動作時に同時減速(CSD)信号自動出力
・同期スタートを行う場合、補間に含まれる軸の同期スタート設定を同一にしておく必要があります。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

例① スタートモード ： FL定速

[VisualC++]

```
TMNETM4Dst dst = {0};              // 移動データ構造体定義, 初期化(0リセット)
TMNETM4Spd spd = {0};              // 速度データ構造体定義, 初期化(0リセット)

dst.Pos[0] = 10000;                // 軸0移動量 ： 10000パルス
dst.Pos[1] = -50000;               // 軸1移動量 ： -50000パルス
dst.StartMode = MNET_STAMODE_FL;   // スタートモード設定

spd.SpdL[0] = 1000;                // 速度 ： 1000PPS
MNETM4LineInc(0, 3, 0x3, &dst, &spd);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst              ’移動データ構造体定義, 初期化(0リセット)
Dim spd As TMNETM4Spd              ’速度データ構造体定義, 初期化(0リセット)

dst.Pos(0) = 10000                 ’軸0移動量 ： 10000パルス
dst.Pos(1) = -50000                ’軸1移動量 ： -50000パルス
dst.StartMode = MNET_STAMODE_FL    ’スタートモード設定

spd.SpdL(0) = 1000                 ’速度 ： 1000PPS
MNETM4LineInc 0, 3, &H3, dst spd
```

[VB.NET]

```
Dim dst As New TMNETM4Dst          ’移動データ構造体定義, 初期化(0リセット)
Dim spd As New TMNETM4Spd          ’速度データ構造体定義, 初期化(0リセット)
dst.Initialize()                   ’構造体初期化
spd.Initialize()                   ’構造体初期化

dst.Pos(0) = 10000                 ’軸0移動量 ： 10000パルス
dst.Pos(1) = -50000                ’軸1移動量 ： -50000パルス
dst.StartMode = MNET_STAMODE_FL    ’スタートモード設定
spd.SpdL(0) = 1000                 ’速度 ： 1000PPS
MNETM4LineInc(0, 3, &H3, dst spd)
```

ライン 0,デバイス 3, 軸 0,1 にスタートモード:FL 定速で Inc 直線補間移動指令を発行します。

Motionnet®
Remote I/O & Remote Motion

例② スタートモード ： 加減速

[VisualC++]

```
TMNETM4Dst dst = {0};              // 移動データ構造体定義, 初期化(0リセット)
TMNETM4Spd spd = {0};              // 速度データ構造体定義, 初期化(0リセット)

dst.Pos[1] =  1000;                // 軸1移動量 ：  1000パルス
dst.Pos[2] = -1000;                // 軸2移動量 ： -1000パルス
dst.Pos[3] = -5000;                // 軸3移動量 ： -5000パルス
dst.StartMode = MNET_STAMODE_UD;   // スタートモード設定

spd.SpdL[0] = 100;                  // FL速度   ：   100PPS
spd.SpdH[0] = 3000;                 // FH速度   ：   3000PPS
spd.AdcMtd[0] = MNET_ADC_BY_TIME;   // 加減速指定方法  ：  時間指定
spd.Smode[0] = MNET_NON_S;          // S字設定  ：   S字なし
spd.Acc[0] = 0.5;                   // 加速時間 ：   0.5秒
spd.Dec[0] = 0.5;                   // 減速時間 ：   0.5秒

MNETM4LineInc(0, 3, 0xE, &dst, &spd);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst              ’移動データ構造体定義, 初期化(0リセット)
Dim spd As TMNETM4Spd              ’速度データ構造体定義, 初期化(0リセット)

dst.Pos(1) =  1000                 ’軸1移動量 ：  1000パルス
dst.Pos(2) = -1000                 ’軸2移動量 ： -1000パルス
dst.Pos(3) = -5000                 ’軸3移動量 ： -5000パルス
dst.StartMode = MNET_STAMODE_ UD   ’スタートモード設定

spd.SpdL(0) = 100                  ’軸0 FL速度   ：   100PPS
spd.SpdH(0) = 3000                 ’軸0 FH速度   ：   3000PPS
spd.AdcMtd(0) = MNET_ADC_BY_TIME   ’軸0 加減速指定方法  ：  時間指定
spd.Smode(0) = MNET_NON_S          ’軸0 S字設定  ：   S字なし
spd.Acc(0) = 0.5                   ’軸0 加速時間 ：   0.5秒
spd.Dec(0) = 0.5                   ’軸0 減速時間 ：   0.5秒

MNETM4LineInc 0, 3, &HE, dst spd
```

[VB.NET]

```
  Dim dst As New TMNETM4Dst             ’移動データ構造体定義, 初期化(0リセット)
  Dim spd As New TMNETM4Spd             ’速度データ構造体定義, 初期化(0リセット)
  dst.Initialize()                      ’構造体初期化
  spd.Initialize()                      ’構造体初期化

  dst.Pos(1) =  1000                    ’軸1移動量 :  1000パルス
  dst.Pos(2) = -1000                    ’軸2移動量 : -1000パルス
  dst.Pos(3) = -5000                    ’軸3移動量 : -5000パルス
  dst.StartMode = MNET_STAMODE_ UD      ’スタートモード設定
  spd.SpdL(0) = 100                     ’軸0 FL速度   :   100PPS
  spd.SpdH(0) = 3000                    ’軸0 FH速度   :   3000PPS
  spd.AdcMtd(0) = MNET_ADC_BY_TIME      ’軸0 加減速指定方法  :  時間指定
  spd.Smode(0) = MNET_NON_S             ’軸0 S字設定  :   S字なし
  spd.Acc(0) = 0.5                      ’軸0 加速時間 :   0.5秒
  spd.Dec(0) = 0.5                      ’軸0 減速時間 :   0.5秒

  MNETM4LineInc(0, 3, &HE, dst spd)
```

ライン 0,デバイス 3, 軸 1,2,3 にスタートモード:加減速で Inc 直線補間移動指令を発行します。

Motionnet®
Remote I/O & Remote Motion

7－1－1－4. MNETM4Ptp

機能

Ptp移動(補間なし絶対位置移動)を起動します。

書式

[VisualC++]

```
short __stdcall MNETM4Ptp(
      short lineNo,
      short devNo,
      short axPtn,
      TMNETM4Dst dst);
```

[VisualBasic]

```
Declare Function MNETM4Ptp Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axPtn As Integer,
      ByRef dst As TMNETM4Dst) As Integer
```

[VB.NET]

```
Declare Function MNETM4Ptp Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axPtn As Short,
      ByRef dst As TMNETM4Dst) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン***
dst	***移動データ構造体*** (7－2－4. 移動データ構造体参照)

備考

速度データはMNETM4SetSpeed関数で設定ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

例① スタートモード : FL定速

[VisualC++]

```
TMNETM4Dst dst = {0};              // 移動データ構造体定義, 初期化(0リセット)

dst.Pos[0] = 1000;                 // 軸0目標位置 : 1000
dst.Pos[1] = -5000;                // 軸1目標位置: -5000
dst.StartMode = MNET_STAMODE_FL;   // スタートモード設定
MNETM4Ptp(0, 3, 0x3, &dst);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst              ’移動データ構造体定義, 初期化(0リセット)

dst.Pos(0) = 1000                  ’軸0目標位置: 1000
dst.Pos(1) = -5000                 ’軸1目標位置: -5000
dst.StartMode = MNET_STAMODE_FL    ’スタートモード設定
MNETM4Ptp 0, 3, &H3, dst
```

[VB.NET]

```
Dim dst As New TMNETM4Dst          ’移動データ構造体定義, 初期化(0リセット)
dst.Initialize()                   ’構造体初期化

dst.Pos(0) = 1000                  ’軸0目標位置: 1000
dst.Pos(1) = -5000                 ’軸1目標位置: -5000
dst.StartMode = MNET_STAMODE_FL    ’スタートモード設定
MNETM4Ptp(0, 3, &H3, dst)
```

ライン0,デバイス3, 軸0,1にスタートモード:FL定速でPtp移動指令を発行します。

例② スタートモード ： 加減速

[VisualC++]

```
TMNETM4Dst dst = {0};              // 移動データ構造体定義, 初期化(0リセット)

dst.Pos[2] = -1000;                // 軸2目標位置: -1000
dst.Pos[3] = -5000;                // 軸3目標位置: -5000
dst.StartMode = MNET_STAMODE_UD;   // スタートモード設定
MNETM4Ptp(0, 3, 0xC, &dst);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst              '移動データ構造体定義, 初期化(0リセット)

dst.Pos(2) = -1000                 ' 軸2目標位置: -1000
dst.Pos(3) = -5000                 ' 軸3目標位置: -5000
dst.StartMode = MNET_STAMODE_ UD   ' スタートモード設定
MNETM4Ptp 0, 3, &HC, dst
```

[VB.NET]

```
Dim dst As New TMNETM4Dst          ' 移動データ構造体定義, 初期化(0リセット)
dst.Initialize()                   ' 構造体初期化

dst.Pos(2) = -1000                 ' 軸2目標位置: -1000
dst.Pos(3) = -5000                 ' 軸3目標位置: -5000
dst.StartMode = MNET_STAMODE_ UD   ' スタートモード設定
MNETM4Ptp(0, 3, &HC, dst)
```

ライン 0,デバイス 3, 軸 2,3 にスタートモード:加減速で Ptp 移動指令を発行します。

Motionnet®
Remote I/O & Remote Motion

7－1－1－5. MNETM4Line

機能

直線補間移動(補間絶対位置移動)を起動します。

書式

[VisualC++]

```
short __stdcall MNETM4Line(
      short lineNo,
      short devNo,
      short axPtn,
      TMNETM4Dst dst
      TMNETM4Spd spd);
```

[VisualBasic]

```
Declare Function MNETM4Line Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axPtn As Integer,
      ByRef dst As TMNETM4Dst,
      ByRef spd As TMNETM4Spd) As Integer
```

[VB.NET]

```
Declare Function MNETM4Line Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axPtn As Short,
      ByRef dst As TMNETM4Dst,
      ByRef spd As TMNETM4Spd) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン***
dst	***移動データ構造体*** (7－2－4. 移動データ構造体参照)
spd	***速度データ構造体*** (9－2. 速度データ構造体 参照)

備考

・動作速度は軸パターンに関わらず、速度データ構造体の軸0のデータに設定します。
・指定する動作速度は全補間軸間のベクトル速度(合成速度)となります。
・直線補間動作関数の実行によりMNETM4SetSpeed関数で設定した、非補間動作用の速度データは未設定状態となります。
・直線補間時は、補間に含まれる軸の以下の設定が強制的に有効となります。
　＊ 同時停止(CSTP)信号による停止。
　＊ 異常停止時、同時停止(CSTP)信号自動出力
　＊ 同時減速(CSD)信号による減速
　＊ 減速時、FL定速動作時に同時減速(CSD)信号自動出力
・同期スタートを行う場合、補間に含まれる軸の同期スタート設定を同一にしておく必要があります。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

例① スタートモード : FL定速

[VisualC++]

```
TMNETM4Dst dst = {0};              // 移動データ構造体定義, 初期化(0リセット)
TMNETM4Spd spd = {0};              // 速度データ構造体定義, 初期化(0リセット)

dst.Pos[0] = 10000;                // 軸0目標位置: 10000
dst.Pos[1] = -50000;               // 軸1目標位置: -50000
dst.StartMode = MNET_STAMODE_FL;   // スタートモード設定

spd.SpdL[0] = 1000;                // 速度 : 1000PPS
MNETM4Line(0, 3, 0x3, &dst, &spd);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst              ’移動データ構造体定義, 初期化(0リセット)
Dim spd As TMNETM4Spd              ’速度データ構造体定義, 初期化(0リセット)

dst.Pos(0) = 10000                 ’軸0目標位置: 10000
dst.Pos(1) = -50000                ’軸1目標位置: -50000
dst.StartMode = MNET_STAMODE_FL    ’スタートモード設定

spd.SpdL(0) = 1000                 ’速度 : 1000PPS
MNETM4Line 0, 3, &H3, dst spd
```

[VB.NET]

```
Dim dst As New TMNETM4Dst          ’移動データ構造体定義, 初期化(0リセット)
Dim spd As New TMNETM4Spd          ’速度データ構造体定義, 初期化(0リセット)
dst.Initialize()                   ’構造体初期化
spd.Initialize()                   ’構造体初期化

dst.Pos(0) = 10000                 ’軸0目標位置: 10000
dst.Pos(1) = -50000                ’軸1目標位置: -50000
dst.StartMode = MNET_STAMODE_FL    ’スタートモード設定
spd.SpdL(0) = 1000                 ’速度 : 1000PPS
MNETM4Line(0, 3, &H3, dst spd)
```

ライン 0,デバイス 3, 軸 0,1 にスタートモード:FL 定速で直線補間移動指令を発行します。

Motionnet®
Remote I/O & Remote Motion

例② スタートモード : 加減速

[VisualC++]

```
TMNETM4Dst dst = {0};              // 移動データ構造体定義, 初期化(0リセット)
TMNETM4Spd spd = {0};              // 速度データ構造体定義, 初期化(0リセット)

dst.Pos[1] =  1000;                // 軸1目標位置:  1000
dst.Pos[2] = -1000;                // 軸2目標位置: -1000
dst.Pos[3] = -5000;                // 軸3目標位置: -5000
dst.StartMode = MNET_STAMODE_UD;   // スタートモード設定

spd.SpdL[0] = 100;                  // FL速度   :   100PPS
spd.SpdH[0] = 3000;                 // FH速度   :   3000PPS
spd.AdcMtd[0] = MNET_ADC_BY_TIME;   // 加減速指定方法  :  時間指定
spd.Smode[0] = MNET_NON_S;          // S字設定  :   S字なし
spd.Acc[0] = 0.5;                   // 加速時間 :   0.5秒
spd.Dec[0] = 0.5;                   // 減速時間 :   0.5秒

MNETM4Line(0, 3, 0xE, &dst, &spd);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst              ' 移動データ構造体定義, 初期化(0リセット)
Dim spd As TMNETM4Spd              ' 速度データ構造体定義, 初期化(0リセット)

dst.Pos(1) =  1000                 ' 軸1目標位置:  1000
dst.Pos(2) = -1000                 ' 軸2目標位置: -1000
dst.Pos(3) = -5000                 '軸3目標位置: -5000
dst.StartMode = MNET_STAMODE_ UD   ' スタートモード設定

spd.SpdL(0) = 100                  ' FL速度   :   100PPS
spd.SpdH(0) = 3000                 ' FH速度   :   3000PPS
spd.AdcMtd(0) = MNET_ADC_BY_TIME   ' 加減速指定方法  :  時間指定
spd.Smode(0) = MNET_NON_S          ' S字設定  :   S字なし
spd.Acc(0) = 0.5                   ' 加速時間 :   0.5秒
spd.Dec(0) = 0.5                   ' 減速時間 :   0.5秒

MNETM4LineInc 0, 3, &HE, dst spd
```


Motionnet®
Remote I/O & Remote Motion

[VB.NET]

```
Dim dst As New TMNETM4Dst              ’移動データ構造体定義, 初期化(0リセット)
Dim spd As New TMNETM4Spd              ’速度データ構造体定義, 初期化(0リセット)
dst.Initialize()                       ’構造体初期化
spd.Initialize()                       ’構造体初期化

dst.Pos(1) =  1000                     ’軸1目標位置:  1000
dst.Pos(2) = -1000                     ’軸2目標位置: -1000
dst.Pos(3) = -5000                     ’軸3目標位置: -5000
dst.StartMode = MNET_STAMODE_ UD       ’スタートモード設定
spd.SpdL(0) = 100                      ’FL速度   :   100PPS
spd.SpdH(0) = 3000                     ’FH速度   :   3000PPS
spd.AdcMtd(0) = MNET_ADC_BY_TIME       ’加減速指定方法  :  時間指定
spd.Smode(0) = MNET_NON_S              ’S字設定  :   S字なし
spd.Acc(0) = 0.5                       ’加速時間 :   0.5秒
spd.Dec(0) = 0.5                       ’減速時間 :   0.5秒

MNETM4LineInc(0, 3, &HE, dst spd)
```

ライン 0,デバイス 3, 軸 1,2,3 にスタートモード:加減速で直線補間移動指令を発行します。

Motionnet®
Remote I/O & Remote Motion

7－1－1－6. MNETM4Continue

機能

連続移動(Stopされるまで動き続ける動作)を起動します。

書式

[VisualC++]

```
short __stdcall MNETM4Continue(
        short lineNo,
        short devNo,
        short axPtn,
        TMNETM4Dst dst);
```

[VisualBasic]

```
Declare Function MNETM4Continue Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal axPtn As Integer,
        ByRef dst As TMNETM4Dst) As Integer
```

[VB.NET]

```
Declare Function MNETM4Continue Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal axPtn As Short,
        ByRef dst As TMNETM4Dst) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン***
dst	***移動データ構造体*** (7－2－4. 移動データ構造体参照)

備考

速度データはMNETM4SetSpeed関数で設定ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

例① スタートモード : FL定速

[VisualC++]

```
TMNETM4Dst dst = {0};                    // 移動データ構造体定義, 初期化(0リセット)

dst.Direction[0] = MNET_DIR_PLUS;        // 軸0動作方向 : プラス
dst.Direction[1] = MNET_DIR_MINUS;       // 軸1動作方向 : マイナス
dst.StartMode = MNET_STAMODE_FL;         // スタートモード設定
MNETM4Continue(0, 3, 0x3, &dst);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst                    ’移動データ構造体定義, 初期化(0リセット)

dst.Direction(0) = MNET_DIR_PLUS         ’軸0動作方向 : プラス
dst.Direction(1) = MNET_DIR_MINUS        ’軸1動作方向 : マイナス
dst.StartMode = MNET_STAMODE_FL          ’スタートモード設定
MNETM4Continue 0, 3, &H3, dst
```

[VB.NET]

```
Dim dst As New TMNETM4Dst                ’移動データ構造体定義, 初期化(0リセット)
dst.Initialize()                         ’構造体初期化

dst.Direction(0) = MNET_DIR_PLUS         ’軸0動作方向 : プラス
dst.Direction(1) = MNET_DIR_MINUS        ’軸1動作方向 : マイナス
dst.StartMode = MNET_STAMODE_FL          ’スタートモード設定
MNETM4Continue(0, 3, &H3, dst)
```

ライン 0,デバイス 3, 軸 0,1 にスタートモード:FL 定速で連続移動指令を発行します。

Motionnet®
Remote I/O & Remote Motion

例② スタートモード ： 加減速

[VisualC++]

```
TMNETM4Dst dst = {0};                  // 移動データ構造体定義, 初期化(0リセット)

dst.Direction [2] = MNET_DIR_PLUS;     // 軸2動作方向 ： プラス
dst.Direction [3] = MNET_DIR_MINUS;    // 軸3動作方向 ： マイナス
dst.StartMode = MNET_STAMODE_UD;       // スタートモード設定
MNETM4Continue (0, 3, 0xC, &dst);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst                  ’移動データ構造体定義, 初期化(0リセット)

dst.Direction (2) = MNET_DIR_PLUS      ’軸2動作方向 ： プラス
dst.Direction (3) = MNET_DIR_MINUS     ’軸3動作方向 ： マイナス
dst.StartMode = MNET_STAMODE_ UD       ’スタートモード設定
MNETM4Continue 0, 3, &HC, dst
```

[VB.NET]

```
Dim dst As New TMNETM4Dst              ’移動データ構造体定義, 初期化(0リセット)
dst.Initialize()                       ’構造体初期化

dst.Direction (2) = MNET_DIR_PLUS      ’軸2動作方向 ： プラス
dst.Direction (3) = MNET_DIR_MINUS     ’軸3動作方向 ： マイナス
dst.StartMode = MNET_STAMODE_ UD       ’スタートモード設定
MNETM4Continue(0, 3, &HC, dst)
```

ライン 0,デバイス 3, 軸 2,3 にスタートモード:加減速で連続移動指令を発行します。

Motionnet®
Remote I/O & Remote Motion

7－1－1－7. MNETM4Org

機能

原点復帰移動を起動します。

書式

[VisualC++]

```
short __stdcall MNETM4Org(
        short lineNo,
        short devNo,
        short axPtn,
        TMNETM4Dst dst);
```

[VisualBasic]

```
Declare Function MNETM4Org Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal axPtn As Integer,
        ByRef dst As TMNETM4Dst) As Integer
```

[VB.NET]

```
Declare Function MNETM4Org Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal axPtn As Short,
        ByRef dst As TMNETM4Dst) As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axPtn	***軸パターン***
dst	***移動データ構造体***　（7－2－4. 移動データ構造体参照）

備考

・速度データはMNETM4SetSpeed関数で設定ください。
・原点復帰モードはMNETM4SetPar関数で設定ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照）

使用例

例① スタートモード ： FL定速

[VisualC++]

```
TMNETM4Dst dst = {0};                  // 移動データ構造体定義, 初期化(0リセット)

dst.Direction[0] = MNET_DIR_PLUS;      // 軸0動作方向 ： プラス
dst.Direction[1] = MNET_DIR_MINUS;     // 軸1動作方向 ： マイナス
dst.StartMode = MNET_STAMODE_FL;       // スタートモード設定
MNETM4Org(0, 3, 0x3, &dst);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst                  ’移動データ構造体定義, 初期化(0リセット)

dst.Direction(0) = MNET_DIR_PLUS       ’軸0動作方向 ： プラス
dst.Direction(1) = MNET_DIR_MINUS      ’軸1動作方向 ： マイナス
dst.StartMode = MNET_STAMODE_FL        ’スタートモード設定
MNETM4Org 0, 3, &H3, dst
```

[VB.NET]

```
Dim dst As New TMNETM4Dst              ’移動データ構造体定義, 初期化(0リセット)
dst.Initialize()                       ’構造体初期化

dst.Direction(0) = MNET_DIR_PLUS       ’軸0動作方向 ： プラス
dst.Direction(1) = MNET_DIR_MINUS      ’軸1動作方向 ： マイナス
dst.StartMode = MNET_STAMODE_FL        ’スタートモード設定
MNETM4Org(0, 3, &H3, dst)
```

ライン 0,デバイス 3, 軸 0,1 にスタートモード:FL 定速で原点復帰移動指令を発行します。

例② スタートモード ： 加減速

[VisualC++]

```
TMNETM4Dst dst = {0};                  // 移動データ構造体定義, 初期化(0リセット)

dst.Direction [2] = MNET_DIR_PLUS;     // 軸2動作方向 ： プラス
dst.Direction [3] = MNET_DIR_MINUS;    // 軸3動作方向 ： マイナス
dst.StartMode = MNET_STAMODE_UD;       // スタートモード設定
MNETM4Org (0, 3, 0xC, &dst);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst                  ’移動データ構造体定義, 初期化(0リセット)

dst.Direction (2) = MNET_DIR_PLUS      ’軸2動作方向 ： プラス
dst.Direction (3) = MNET_DIR_MINUS     ’軸3動作方向 ： マイナス
dst.StartMode = MNET_STAMODE_ UD       ’スタートモード設定
MNETM4Org 0, 3, &HC, dst
```

[VB.NET]

```
Dim dst As New TMNETM4Dst              ’移動データ構造体定義, 初期化(0リセット)
dst.Initialize()                       ’構造体初期化

dst.Direction (2) = MNET_DIR_PLUS      ’軸2動作方向 ： プラス
dst.Direction (3) = MNET_DIR_MINUS     ’軸3動作方向 ： マイナス
dst.StartMode = MNET_STAMODE_ UD       ’スタートモード設定
MNETM4Org(0, 3, &HC, dst)
```

ライン 0,デバイス 3, 軸 2,3 にスタートモード:加減速で原点復帰移動指令を発行します。

Motionnet®
Remote I/O & Remote Motion

7－1－1－8. MNETM4SetSpeedNo

機能

速度データ設定 （軸番号指定）

書式

[VisualC++]

```
short __stdcall MNETM4SetSpeedNo(
        short lineNo,
        short devNo,
        short axNo,
        TMNETM4Spd* spd,
        short staMode);
```

[VisualBasic]

```
Declare Function MNETM4SetSpeedNo Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal axNo As Integer,
        ByRef spd As TMNETM4Spd,
        ByVal staMode As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM4SetSpeedNo Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal axNo As Short,
        ByRef spd As TMNETM4Spd,
        ByVal staMode As Short) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

axNo ***軸番号*** (0始まり)

spd ***速度データ構造体*** （7－2－3. 速度データ構造体参照）

staMode ***スタートモード***

識別子	値	内容
MNET_STAMODE_FL	0050h	FL定速
MNET_STAMODE_FH	0051h	FH 定速
MNET_STAMODE_UD	0053h	加減速

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照）

使用例

例① スタートモード ： FL定速

[VisualC++]

```
TMNETM4Spd spd = {0};              // 速度データ構造体定義, 初期化(0リセット)

spd.SpdL[0] = 100;                 // 軸0 FL速度   :   100PPS
MNETM4SetSpeedNo(0, 3, 0, &spd, MNET_STAMODE_FL);
```

[VisualBasic]

```
Dim spd As TMNETM4Spd              ’速度データ構造体定義, 初期化(0リセット)

spd.SpdL(0) = 100                  ’軸0 FL速度   :   100PPS
MNETM4SetSpeedNo 0, 3, 0, spd, MNET_STAMODE_FL
```

[VB.NET]

```
Dim spd As New TMNETM4Spd          ’速度データ構造体定義, 初期化(0リセット)
spd.Initialize()                   ’構造体初期化

spd.SpdL(0) = 100                  ’軸0 FL速度   :   100PPS
MNETM4SetSpeedNo(0, 3, 0, spd, MNET_STAMODE_FL)
```

スタートモード:FL 定速でライン 0,デバイス 3 の軸 0 の速度を設定します。

Motionnet®
Remote I/O & Remote Motion

例② スタートモード : 加減速

[VisualC++]

```
TMNETM4Spd spd = {0};                  // 速度データ構造体定義, 初期化(0リセット)

spd.SpdL[2] = 2000;                    // 軸2 FL速度   :   2000PPS
spd.SpdH[2] = 10000;                   // 軸2 FH速度   :   10000PPS
spd.AdcMtd[2] = MNET_ADC_BY_RATE;      // 軸2 加減速指定方法  :  レート指定
spd.Smode[2] = MNET_USE_S;             // 軸2 S字設定   :   S字
spd.Acc[2] = 1000;                     // 軸2 加速レート:   1000PPS/秒
spd.Dec[2] = 5000;                     // 軸2 減速レート:   5000PPS/秒

MNETM4SetSpeedNo(0, 3, 2, &spd, MNET_STAMODE_UD);
```

[VisualBasic]

```
Dim spd As TMNETM4Spd                  ’速度データ構造体定義, 初期化(0リセット)

spd.SpdL(2) = 2000                     ’軸2 FL速度   :   2000PPS
spd.SpdH(2) = 10000                    ’軸2 FH速度   :   10000PPS
spd.AdcMtd(2) = MNET_ADC_BY_RATE       ’軸2 加減速指定方法  :  レート指定
spd.Smode(2) = MNET_USE_S              ’軸2 S字設定   :   S字
spd.Acc(2) = 1000                      ’軸2 加速レート:   1000PPS/秒
spd.Dec(2) = 5000                      ’軸2 減速レート:   5000PPS/秒

MNETM4SetSpeedNo 0, 3, 2, spd, MNET_STAMODE_UD
```

[VB.NET]

```
Dim spd As New TMNETM4Spd              ’速度データ構造体定義, 初期化(0リセット)
spd.Initialize()                       ’構造体初期化

spd.SpdL(2) = 2000                     ’軸2 FL速度   :   2000PPS
spd.SpdH(2) = 10000                    ’軸2 FH速度   :   10000PPS
spd.AdcMtd(2) = MNET_ADC_BY_RATE       ’軸2 加減速指定方法  :  レート指定
spd.Smode(2) = MNET_USE_S              ’軸2 S字設定   :   S字
spd.Acc(2) = 1000                      ’軸2 加速レート:   1000PPS/秒
spd.Dec(2) = 5000                      ’軸2 減速レート:   5000PPS/秒

MNETM4SetSpeedNo(0, 3, 2, spd, MNET_STAMODE_UD)
```

スタートモード:加減速でライン 0,デバイス 3 の軸 2 の速度を設定します。

Motionnet®
Remote I/O & Remote Motion

7－1－1－9. MNETM4PtpIncNo

機能

PtpInc移動(補間なし相対位置移動)を起動します。 (軸番号指定)

書式

[VisualC++]

```
short __stdcall MNETM4PtpIncNo(
        short lineNo,
        short devNo,
        short axNo,
        TMNETM4Dst dst);
```

[VisualBasic]

```
Declare Function MNETM4PtpIncNo Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal axNo As Integer,
        ByRef dst As TMNETM4Dst) As Integer
```

[VB.NET]

```
Declare Function MNETM4PtpIncNo Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal axNo As Short,
        ByRef dst As TMNETM4Dst) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
dst	***移動データ構造体*** (7－2－4. 移動データ構造体参照)

備考

速度データはMNETM4SetSpeed関数で設定ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

例① スタートモード ： FL定速

[VisualC++]

```
TMNETM4Dst dst = {0};              // 移動データ構造体定義, 初期化(0リセット)

dst.Pos[1] = -5000;                // 軸1移動量 ： -5000パルス
dst.StartMode = MNET_STAMODE_FL;   // スタートモード設定
MNETM4PtpIncNo(0, 3, 1, &dst);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst              ’移動データ構造体定義, 初期化(0リセット)

dst.Pos(1) = -5000                 ’軸1移動量 ： -5000パルス
dst.StartMode = MNET_STAMODE_FL    ’スタートモード設定
MNETM4PtpIncNo 0, 3, 1, dst
```

[VB.NET]

```
Dim dst As New TMNETM4Dst          ’移動データ構造体定義, 初期化(0リセット)
dst.Initialize()                   ’構造体初期化

dst.Pos(1) = -5000                 ’軸1移動量 ： -5000パルス
dst.StartMode = MNET_STAMODE_FL    ’スタートモード設定
MNETM4PtpIncNo(0, 3, 1, dst)
```

ライン 0,デバイス 3, 軸 1 にスタートモード:FL 定速で PtpInc 移動指令を発行します。

Motionnet®
Remote I/O & Remote Motion

例② スタートモード ： 加減速

[VisualC++]

```
TMNETM4Dst dst = {0};              // 移動データ構造体定義, 初期化(0リセット)

dst.Pos[3] = -5000;                // 軸3移動量 : -5000パルス
dst.StartMode = MNET_STAMODE_UD;   // スタートモード設定
MNETM4PtpIncNo(0, 3, 3, &dst);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst              ’移動データ構造体定義, 初期化(0リセット)

dst.Pos(3) = -5000                 ’軸3移動量 : -5000パルス
dst.StartMode = MNET_STAMODE_ UD   ’スタートモード設定
MNETM4PtpIncNo 0, 3, 3, dst
```

[VB.NET]

```
Dim dst As New TMNETM4Dst          ’移動データ構造体定義, 初期化(0リセット)
dst.Initialize()                   ’構造体初期化

dst.Pos(3) = -5000                 ’軸3移動量 : -5000パルス
dst.StartMode = MNET_STAMODE_ UD   ’スタートモード設定
MNETM4PtpIncNo(0, 3, 3, dst)
```

ライン 0,デバイス 3, 軸 3 にスタートモード:加減速で PtpInc 移動指令を発行します。

Motionnet®
Remote I/O & Remote Motion

7－1－1－10. MNETM4PtpNo

機能

Ptp移動(補間なし絶対位置移動)を起動します。 (軸番号指定)

書式

[VisualC++]

```
short __stdcall MNETM4PtpNo(
      short lineNo,
      short devNo,
      short axNo,
      TMNETM4Dst dst);
```

[VisualBasic]

```
Declare Function MNETM4PtpNo Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axNo As Integer,
      ByRef dst As TMNETM4Dst) As Integer
```

[VB.NET]

```
Declare Function MNETM4PtpNo Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axNo As Short,
      ByRef dst As TMNETM4Dst) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
dst	***移動データ構造体*** (7－2－4. 移動データ構造体参照)

備考

速度データはMNETM4SetSpeed関数で設定ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

例① スタートモード : FL定速

[VisualC++]

```
TMNETM4Dst dst = {0};              // 移動データ構造体定義, 初期化(0リセット)

dst.Pos[0] = 1000;                 // 軸0目標位置: 1000
dst.StartMode = MNET_STAMODE_FL;   // スタートモード設定
MNETM4PtpNo(0, 3, 0, &dst);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst              ’移動データ構造体定義, 初期化(0リセット)

dst.Pos(0) = 1000                  ’軸0目標位置: 1000
dst.StartMode = MNET_STAMODE_FL    ’スタートモード設定
MNETM4PtpNo 0, 3, 0, dst
```

[VB.NET]

```
Dim dst As New TMNETM4Dst          ’移動データ構造体定義, 初期化(0リセット)
dst.Initialize()                   ’構造体初期化

dst.Pos(0) = 1000                  ’軸0目標位置: 1000
dst.StartMode = MNET_STAMODE_FL    ’スタートモード設定
MNETM4PtpNo(0, 3, 0, dst)
```

ライン 0,デバイス 3, 軸 0 にスタートモード:FL 定速で Ptp 移動指令を発行します。

Motionnet®
Remote I/O & Remote Motion

例② スタートモード ： 加減速

[VisualC++]

```
TMNETM4Dst dst = {0};              // 移動データ構造体定義, 初期化(0リセット)

dst.Pos[3] = -5000;                // 軸3目標位置: -5000
dst.StartMode = MNET_STAMODE_UD;   // スタートモード設定
MNETM4PtpNo(0, 3, 3, &dst);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst              ’移動データ構造体定義, 初期化(0リセット)

dst.Pos(3) = -5000                 ’軸3目標位置 ： -5000
dst.StartMode = MNET_STAMODE_ UD   ’スタートモード設定
MNETM4PtpNo 0, 3, 3, dst
```

[VB.NET]

```
Dim dst As New TMNETM4Dst          ’移動データ構造体定義, 初期化(0リセット)
dst.Initialize()                   ’構造体初期化

dst.Pos(3) = -5000                 ’軸3目標位置 ： -5000
dst.StartMode = MNET_STAMODE_ UD   ’スタートモード設定
MNETM4PtpNo(0, 3, 3, dst)
```

ライン 0,デバイス 3, 軸 3 にスタートモード:加減速で Ptp 移動指令を発行します。

Motionnet®
Remote I/O & Remote Motion

7－1－1－11. MNETM4ContinueNo

機能

連続移動(Stopされるまで動き続ける動作)を起動します。 (軸番号指定)

書式

[VisualC++]

```
short __stdcall MNETM4ContinueNo(
        short lineNo,
        short devNo,
        short axNo,
        TMNETM4Dst dst);
```

[VisualBasic]

```
Declare Function MNETM4ContinueNo Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal axNo As Integer,
        ByRef dst As TMNETM4Dst) As Integer
```

[VB.NET]

```
Declare Function MNETM4ContinueNo Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal axNo As Short,
        ByRef dst As TMNETM4Dst) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
dst	***移動データ構造体*** (7－2－4. 移動データ構造体参照)

備考

速度データはMNETM4SetSpeed関数で設定ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

例① スタートモード ： FL定速

[VisualC++]

```
TMNETM4Dst dst = {0};                    // 移動データ構造体定義, 初期化(0リセット)

dst.Direction[0] = MNET_DIR_PLUS;        // 軸0動作方向 : プラス
dst.StartMode = MNET_STAMODE_FL;         // スタートモード設定
MNETM4ContinueNo(0, 3, 0, &dst);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst                    ’移動データ構造体定義, 初期化(0リセット)

dst.Direction(0) = MNET_DIR_PLUS         ’軸0動作方向 : プラス
dst.StartMode = MNET_STAMODE_FL          ’スタートモード設定
MNETM4ContinueNo 0, 3, 0, dst
```

[VB.NET]

```
Dim dst As New TMNETM4Dst                ’移動データ構造体定義, 初期化(0リセット)
dst.Initialize()                         ’構造体初期化

dst.Direction(0) = MNET_DIR_PLUS         ’軸0動作方向 : プラス
dst.StartMode = MNET_STAMODE_FL          ’スタートモード設定
MNETM4ContinueNo(0, 3, 0, dst)
```

ライン 0,デバイス 3, 軸 0 にスタートモード:FL 定速で連続移動指令を発行します。

Motionnet®
Remote I/O & Remote Motion

例② スタートモード ： 加減速

[VisualC++]

```
TMNETM4Dst dst = {0};                  // 移動データ構造体定義, 初期化(0リセット)

dst.Direction [2] = MNET_DIR_PLUS;     // 軸2動作方向 ： プラス
dst.StartMode = MNET_STAMODE_UD;       // スタートモード設定
MNETM4ContinueNo (0, 3, 2, &dst);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst                  ’移動データ構造体定義, 初期化(0リセット)

dst.Direction (2) = MNET_DIR_PLUS      ’軸2動作方向 ： プラス
dst.StartMode = MNET_STAMODE_ UD       ’スタートモード設定
MNETM4ContinueNo 0, 3, 2, dst
```

[VB.NET]

```
Dim dst As New TMNETM4Dst              ’移動データ構造体定義, 初期化(0リセット)
dst.Initialize()                       ’構造体初期化

dst.Direction (2) = MNET_DIR_PLUS      ’軸2動作方向 ： プラス
dst.StartMode = MNET_STAMODE_ UD       ’スタートモード設定
MNETM4ContinueNo(0, 3, 2, dst)
```

ライン 0,デバイス 3, 軸 2 にスタートモード:加減速で連続移動指令を発行します。

Motionnet®
Remote I/O & Remote Motion

7－1－1－12. MNETM4OrgNo

機能

原点復帰移動を起動します。 (軸番号指定)

書式

[VisualC++]

```
short __stdcall MNETM4OrgNo(
        short lineNo,
        short devNo,
        short axNo,
        TMNETM4Dst dst);
```

[VisualBasic]

```
Declare Function MNETM4OrgNo Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal axNo As Integer,
        ByRef dst As TMNETM4Dst) As Integer
```

[VB.NET]

```
Declare Function MNETM4OrgNo Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal axNo As Short,
        ByRef dst As TMNETM4Dst) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
dst	***移動データ構造体*** (7－2－4. 移動データ構造体参照)

備考

・速度データはMNETM4SetSpeed関数で設定ください。
・原点復帰モードはMNETM4SetPar関数で設定ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

Motionnet® Remote I/O & Remote Motion

使用例

例① スタートモード ： FL定速

[VisualC++]

```
TMNETM4Dst dst = {0};                    // 移動データ構造体定義, 初期化(0リセット)

dst.Direction[0] = MNET_DIR_PLUS;        // 軸0動作方向 : プラス
dst.StartMode = MNET_STAMODE_FL;         // スタートモード設定
MNETM4OrgNo(0, 3, 0, &dst);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst                    ’移動データ構造体定義, 初期化(0リセット)

dst.Direction(0) = MNET_DIR_PLUS         ’軸0動作方向 : プラス
dst.StartMode = MNET_STAMODE_FL          ’スタートモード設定
MNETM4OrgNo 0, 3, 0, dst
```

[VB.NET]

```
Dim dst As New TMNETM4Dst                ’移動データ構造体定義, 初期化(0リセット)
dst.Initialize()                         ’構造体初期化

dst.Direction(0) = MNET_DIR_PLUS         ’軸0動作方向 : プラス
dst.StartMode = MNET_STAMODE_FL          ’スタートモード設定
MNETM4OrgNo(0, 3, 0, dst)
```

ライン 0,デバイス 3, 軸 0 にスタートモード:FL 定速で原点復帰移動指令を発行します。

例② スタートモード ： 加減速

[VisualC++]

```
TMNETM4Dst dst = {0};                  // 移動データ構造体定義, 初期化(0リセット)

dst.Direction [2] = MNET_DIR_PLUS;     // 軸2動作方向 ： プラス
dst.StartMode = MNET_STAMODE_UD;       // スタートモード設定
MNETM4OrgNo (0, 3, 2, &dst);
```

[VisualBasic]

```
Dim dst As TMNETM4Dst                  ’移動データ構造体定義, 初期化(0リセット)

dst.Direction (2) = MNET_DIR_PLUS      ’軸2動作方向 ： プラス
dst.StartMode = MNET_STAMODE_ UD       ’スタートモード設定
MNETM4OrgNo 0, 3, 2, dst
```

[VB.NET]

```
Dim dst As New TMNETM4Dst              ’移動データ構造体定義, 初期化(0リセット)
dst.Initialize()                       ’構造体初期化

dst.Direction (2) = MNET_DIR_PLUS      ’軸2動作方向 ： プラス
dst.StartMode = MNET_STAMODE_ UD       ’スタートモード設定
MNETM4OrgNo(0, 3, 2, dst)
```

ライン 0,デバイス 3, 軸 2 にスタートモード:加減速で原点復帰移動指令を発行します。

Motionnet®
Remote I/O & Remote Motion

7－1－1－13. MNETM4StopChk

機能

軸停止確認

書式

[VisualC++]

```
short __stdcall MNETM4StopChk(
      short lineNo,
      short devNo,
      short axPtn,
      short* stopSts);
```

[VisualBasic]

```
Declare Function MNETM4StopChk Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axPtn As Integer,
      ByRef stopSts As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM4StopChk Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axPtn As Short,
      ByRef stopSts As Short) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)	
devNo	***デバイス番号*** (0始まり)	
axPtn	***軸パターン***	
stopSts	***軸状態***	0 ： 動作中軸あり
		1 ： 全軸正常停止
		-1 ： 異常停止軸有り
		-2 ： その他のエラー(指定値不正,通信エラーなど)により状態取得失敗

説明

・指定した軸パターンの軸の動作状態が取得できます。
・軸パターンに含まれる軸番号の小さい軸から状態を確認されます。
・異常停止軸(エラー割込要因にデータがある軸)が見つかった場合、その時点で確認は中断されます。
・同時スタート(CSTA)信号他の待ち状態は、動作中として扱われます。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

[VisualC++]

```
short stopSts;

MNETM4StopChk(0, 3, 0x3, &stopSts);
```

[VisualBasic]

```
Dim stopSts As Integer

MNETM4StopChk 0, 3, &H3, stopSts
```

[VB.NET]

```
Dim stopSts As Short

MNETM4StopChk(0, 3, &H3, stopSts)
```

ライン 0,デバイス 3, 軸パターン 3h(軸 0,1)の停止を確認します。

7－1－1－14. MNETM4GetREST

機能

エラー要因取得

書式

[VisualC++]

```
short __stdcall MNETM4GetREST(
      short lineNo,
      short devNo,
      short axNo,
      long* rest);
```

[VisualBasic]

```
Declare Function MNETM4GetREST Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axNo As Integer,
      ByRef rest As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM4GetREST Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axNo As Short,
      ByRef rest As Integer) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
rest	**データ格納**

説明

・指定した軸のエラー要因データを取得できます。(7－3－2－1. エラー割り込み要因参照)
・このデータは読出しによりリセットされます。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

[VisualC++]

```
long rest;

MNETM4GetREST(0, 3, 0, &rest);
```

[VisualBasic]

```
Dim rest As Long

MNETM4GetREST 0, 3, 0, rest
```

[VB.NET]

```
Dim rest As Long

MNETM4GetREST(0, 3, 0, rest)
```

ライン 0,デバイス 3, 軸 0 のエラー要因データを取得します。

7－1－1－15. MNETM4GetRIST

機能

イベント要因取得

書式

[VisualC++]

```
short __stdcall MNETM4GetRIST(
      short lineNo,
      short devNo,
      short axNo,
      long* rist);
```

[VisualBasic]

```
Declare Function MNETM4GetRIST Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axNo As Integer,
      ByRef rist As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM4GetRIST Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axNo As Short,
      ByRef rist As Integer) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
rist	**データ格納**

説明

・指定した軸のイベント要因データを取得できます。(7－3－2－2. イベント割り込み要因参照)
・このデータは読出しによりリセットされます。
・イベントの有効無効設定はMNETM4SetPar関数で行ないます。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

[VisualC++]

```
long rist;

MNETM4GetRIST(0, 3, 0, &rist);
```

[VisualBasic]

```
Dim rist As Long

MNETM4GetRIST 0, 3, 0, rist
```

[VB.NET]

```
Dim rist As Integer

MNETM4GetRIST(0, 3, 0, rist)
```

ライン 0,デバイス 3, 軸 0 のイベント要因データを取得します。

7－1－1－16. MNETM4CancelPreReg

機能

次動作キャンセル

書式

[VisualC++]

```
short __stdcall MNETM4CancelPreReg(
      short lineNo,
      short devNo,
      short axPtn);
```

[VisualBasic]

```
Declare Function MNETM4CancelPreReg Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axPtn As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM4CancelPreReg Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axPtn As Short) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン***

説明

指定した軸パターンに設定されている次動作をキャンセルします。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

[VisualC++]

```
MNETM4CancelPreReg(0, 3, 0x3);
```

[VisualBasic]

```
MNETM4CancelPreReg 0, 3, &H3
```

[VB.NET]

```
MNETM4CancelPreReg(0, 3, &H3)
```

ライン 0,デバイス 3, 軸パターン 3h(軸 0,1)に設定された次動作をキャンセルします。

7－1－1－17. MNETM4Stop

機能

軸停止

書式

[VisualC++]

```
short __stdcall MNETM4Stop(
      short lineNo,
      short devNo,
      short axPtn,
      short stopMode);
```

[VisualBasic]

```
Declare Function MNETM4Stop Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axPtn As Integer,
      ByVal stopMode As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM4Stop Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axPtn As Short,
      ByVal stopMode As Short) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

axPtn ***軸パターン***

stopMode **停止モード指定**

識別子	値	内容
MNET_STOP	0	即停止
MNET_SDSTOP	1	減速停止
MNET_EMGSTOP	2	非常停止

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

[VisualC++]

```
MNETM4Stop(0, 3, 0x3, MNET_SDSTOP);
```

[VisualBasic]

```
MNETM4Stop 0, 3, &H3, MNET_SDSTOP
```

[VB.NET]

```
MNETM4Stop(0, 3, &H3, MNET_SDSTOP)
```

ライン 0,デバイス 3, 軸パターン 3h(軸 0,1)を減速停止します。

7－1－1－18. MNETM4SetBitData

機能

ビットデータ設定

書式

[VisualC++]

```
short __stdcall MNETM4SetBitData(
        short lineNo,
        short devNo,
        short axNo,
        short sel,
        short data);
```

[VisualBasic]

```
Declare Function MNETM4SetBitData Lib "PPCI_L112.dll" (
        ByVal lineNo As Integer,
        ByVal devNo As Integer,
        ByVal axNo As Integer,
        ByVal sel As Integer,
        ByVal data As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM4SetBitData Lib "PPCI_L112.dll" (
        ByVal lineNo As Short,
        ByVal devNo As Short,
        ByVal axNo As Short,
        ByVal sel As Short,
        ByVal data As Short) As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axNo	***軸番号***（0始まり）
sel	***対象選択*** （7－3－3－9. ビット選択参照）Set可能なビット以外は指定できません。
data	***設定値*** 1: ON, 0:OFF, -1:反転

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照）

使用例

[VisualC++]

```
MNETM4SetBitData(0, 3, 0, MNETM4_SEL_BIT_P0, 1);
```

[VisualBasic]

```
MNETM4SetBitData 0, 3, 0, MNETM4_SEL_BIT_P0, 1
```

[VB.NET]

```
MNETM4SetBitData(0, 3, 0, MNETM4_SEL_BIT_P0, 1)
```

ライン 0, デバイス 3,軸 0 のビット P0 に 1 を設定します。

7－1－1－19. MNETM4GetBitData

機能

ビットデータ取得

書式

[VisualC++]

```
short __stdcall MNETM4GetBitData(
      short lineNo,
      short devNo,
      short axNo,
      short sel,
      short* data);
```

[VisualBasic]

```
Declare Function MNETM4GetBitData Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axNo As Integer,
      ByVal sel As Integer,
      ByRef data As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM4GetBitData Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axNo As Short,
      ByVal sel As Short,
      ByRef data As Short) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
sel	***対象選択*** (7－3－3－9. ビット選択参照)参照
data	***取得データ格納変数***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照）

使用例

[VisualC++]

```
short data;

MNETM4GetBitData(0, 3, 0, MNETM4_SEL_BIT_P0, &data);
```

[VisualBasic]

```
Dim data As Integer

MNETM4GetBitData 0, 3, 0, MNETM4_SEL_BIT_P0, data
```

[VB.NET]

```
Dim data As Short

MNETM4GetBitData(0, 3, 0, MNETM4_SEL_BIT_P0, data)
```

ライン 0, デバイス 3, 軸 0 のビット P0 の状態を変数 data に取得します。

7－1－1－20. MNETM4GetBitDataStruct

機能

ビットデータ読出し(構造体に一括)

書式

[VisualC++]

```
short __stdcall MNETM4GetBitDataStruct(
      short lineNo,
      short devNo,
      short axNo,
      TMNETM4BitData*  bit);
```

[VisualBasic]

```
Declare Function MNETM4GetBitDataStruct Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axNo As Integer,
      ByRef bit As TMNETM4BitData) As Integer
```

[VB.NET]

```
Declare Function MNETM4GetBitDataStruct Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axNo As Short,
      ByRef bit As TMNETM4BitData) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
bit	***ビットデータ構造体*** (7－3－1. ビットデータ構造体) 参照

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

[VisualC++]

```
TMNETM4BitData bit = {0};                // ビットデータ構造体定義

MNETM4GetBitDataStruct(0, 3, 0, &bit);  // ライン0, デバイス3, 軸0のビットデータを一括取得
if (bit.Spel != 0)                       // Spel(プラスEL)ビットによる判定
  m_pel.SetCheck(TRUE);                  // ON時:チェックON
else
  m_pel.SetCheck(FALSE);                 // OFF時: チェックOFF

if (bit.Salm != 0)                       // Salm(ALM)ビットによる判定
  m_alm.SetCheck(TRUE);                  // ON時: チェックON
else
  m_alm.SetCheck(FALSE);                 // OFF時: チェックOFF
```

[VisualBasic]

```
Dim bit As TMNETM4BitData                ' ビットデータ構造体定義

MNETM4GetBitDataStruct 0, 3, 0, bit      ' ライン0, デバイス3,軸0のビットデータを一括取得
If (bit.Spel <> 0) Then                  ' Spel(プラスEL)ビットによる判定
  ShpPEL.FillColor = &HFF&               ' ON時:明赤色
Else
  ShpPEL.FillColor = &H80&               ' OFF時:暗赤色
End If

If (bit.Salm <> 0) Then                  ' Salm(ALM)ビットによる判定
  ShpAlm.FillColor = &HFF&               ' ON時:明赤色
Else
  ShpAlm.FillColor = &H80&               ' OFF時:暗赤色
End If
```

[VB.NET]

```
Dim bit As TMNETM4BitData                ' ビットデータ構造体定義

MNETM4GetBitDataStruct(0, 3, 0, bit)     ' ライン0, デバイス3,軸0のビットデータを一括取得
If (bit.Spel <> 0) Then                  ' Spel(プラスEL)ビットによる判定
  ShpPEL.BackColor = System.Drawing.Color.Red          ' ON時:明赤色
Else
  ShpPEL.BackColor = System.Drawing.Color.DarkRed      ' OFF時:暗赤色
End If

If (bit.Salm <> 0) Then                  ' Salm(ALM)ビットによる判定
  ShpAlm.BackColor = System.Drawing.Color.Red          ' ON時:明赤色
Else
  ShpAlm.BackColor = System.Drawing.Color.DarkRed      ' OFF時:暗赤色
End If
```

7－1－1－21. MNETM4GetSpeed

機能

速度データ取得

書式

[VisualC++]

```
short __stdcall MNETM4GetSpeed(
      short lineNo,
      short devNo,
      short axNo,
      double* speed);
```

[VisualBasic]

```
Declare Function MNETM4GetSpeed Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axNo As Integer,
      ByRef speed As Double) As Integer
```

[VB.NET]

```
Declare Function MNETM4GetSpeed Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axNo As Short,
      ByRef speed As Double) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
speed	***取得データ格納変数***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

[VisualC++]

```
double speed;

MNETM4GetSpeed(0, 3, 0, &speed);
```

[VisualBasic]

```
Dim speed As Double

MNETM4GetSpeed 0, 3, 0, speed
```

[VB.NET]

```
Dim speed As Double

MNETM4GetSpeed(0, 3, 0, speed)
```

ライン 0, デバイス 3,軸 0 の速度データを変数 speed に取得します。

7－1－1－22. MNETM4GetCounter

機能

カウンタ値取得

書式

[VisualC++]

```
short __stdcall MNETM4GetCounter(
      short lineNo,
      short devNo,
      short axNo,
      short selCnt,
      long* data);
```

[VisualBasic]

```
Declare Function MNETM4GetCounter Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axNo As Integer,
      ByVal selCnt As Integer,
      ByRef data As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM4GetCounter Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axNo As Short,
      ByVal selCnt As Short,
      ByRef data As Integer) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

axNo ***軸番号*** (0始まり)

selCnt ***対象カウンタ選択***

識別子	値	内容
MNET_SEL_COUNTER1	0	カウンタ1
MNET_SEL_COUNTER2	1	カウンタ2
MNET_SEL_COUNTER3	2	カウンタ3

data ***取得データ格納変数***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

[VisualC++]

```
long data;

MNETM4GetCounter(0, 3, 0, MNET_SEL_COUNTER1, &data);
```

[VisualBasic]

```
Dim data As Long

MNETM4GetCounter 0, 3, 0, MNET_SEL_COUNTER1, data
```

[VB.NET]

```
Dim data As Integer

MNETM4GetCounter(0, 3, 0, MNET_SEL_COUNTER1, data)
```

ライン 0, デバイス 3,軸 0 のカウンタ 1 のデータを変数 data に取得します。

7－1－1－23. MNETM4SetCounter

機能

カウンタ値設定

書式

[VisualC++]

```
short __stdcall MNETM4SetCounter(
      short lineNo,
      short devNo,
      short axNo,
      short selCnt,
      long data);
```

[VisualBasic]

```
Declare Function MNETM4SetCounter Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axNo As Integer,
      ByVal selCnt As Integer,
      ByVal data As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM4SetCounter Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axNo As Short,
      ByVal selCnt As Short,
      ByVal data As Integer) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

axNo ***軸番号*** (0始まり)

selCnt ***対象カウンタ選択***

識別子	値	内容
MNET_SEL_COUNTER1	0	カウンタ1
MNET_SEL_COUNTER2	1	カウンタ2
MNET_SEL_COUNTER3	2	カウンタ3

data ***設定値***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

[VisualC++]

```
MNETM4SetCounter(0, 3, 0, MNET_SEL_COUNTER1, 200);
```

[VisualBasic]

```
MNETM4SetCounter 0, 3, 0, MNET_SEL_COUNTER1, 200
```

[VB.NET]

```
MNETM4SetCounter(0, 3, 0, MNET_SEL_COUNTER1, 200)
```

ライン 0, デバイス 3,軸 0 のカウンタ 1 にデータ:200 を設定します。

7－1－1－24. MNETM4GetLtc

機能

ラッチデータ取得

書式

[VisualC++]

```
short __stdcall MNETM4GetLtc(
      short lineNo,
      short devNo,
      short axNo,
      short sel,
      long* data);
```

[VisualBasic]

```
Declare Function MNETM4GetLtc Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axNo As Integer,
      ByVal sel As Integer,
      ByRef data As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM4GetLtc Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axNo As Short,
      ByVal sel As Short,
      ByRef data As Integer) As Short
```

パラメータ

lineNo ***ライン番号*** (0始まり)

devNo ***デバイス番号*** (0始まり)

axNo ***軸番号*** (0始まり)

sel ***対象ラッチ選択***

識別子	値	内容
MNET_SEL_LTC1	0	ラッチ1
MNET_SEL_LTC2	1	ラッチ2
MNET_SEL_LTC3	2	ラッチ3

data ***取得データ格納変数***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

[VisualC++]

```
long data;

MNETM4GetLtc(0, 3, 0, MNET_SEL_LTC1, &data);
```

[VisualBasic]

```
Dim data As Long

MNETM4GetLtc 0, 3, 0, MNET_SEL_LTC1, data
```

[VB.NET]

```
Dim data As Integer

MNETM4GetLtc(0, 3, 0, MNET_SEL_LTC1, data)
```

ライン 0, デバイス 3,軸 0 のラッチ 1 のデータを変数 data に取得します。

7－1－1－25. MNETM4SoftLTC

機能

LTC入力代行コマンド発行

書式

[VisualC++]

```
short __stdcall MNETM4SoftLTC(
      short lineNo,
      short devNo
      short axPtn);
```

[VisualBasic]

```
Declare Function MNETM4SoftLTC Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axPtn As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM4SoftLTC Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axPtn As Short) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン***

備考

ご利用にあたっては(7－2－17. ラッチ)の項を参照ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照）

使用例

[VisualC++]

```
MNETM4SoftLTC(0, 3, 0x3);
```

[VisualBasic]

```
MNETM4SoftLTC 0, 3, &H3
```

[VB.NET]

```
MNETM4SoftLTC(0, 3, &H3)
```

ライン0, デバイス3,軸パターン3h(軸0,1)にLTC入力代行コマンドを発行します。

Motionnet® Remote I/O & Remote Motion

7－1－1－26. MNETM4SoftPCS

機能

PCS入力代行コマンド発行

書式

[VisualC++]

```
short __stdcall MNETM4SoftPCS(
      short lineNo,
      short devNo,
      short axPtn);
```

[VisualBasic]

```
Declare Function MNETM4SoftPCS Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axPtn As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM4SoftPCS Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axPtn As Short) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン***

備考

ご利用にあたっては(7－2－11. PCS 入力)の項を参照ください。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照）

使用例

[VisualC++]

```
MNETM4SoftPCS(0, 3, 0x3);
```

[VisualBasic]

```
MNETM4SoftPCS 0, 3, &H3
```

[VB.NET]

```
MNETM4SoftPCS(0, 3, &H3)
```

ライン0, デバイス3,軸パターン3h(軸0,1)にPCS入力代行コマンドを発行します。

7－1－1－27. MNETM4OutCsta

機能

同時スタート(CSTA)信号出力

書式

[VisualC++]

```
short __stdcall MNETM4OutCsta(
      short lineNo,
      short devNo,
      short axPtn,
      short cstaMode);
```

[VisualBasic]

```
Declare Function MNETM4OutCsta Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axPtn As Integer,
      ByVal cstaMode As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM4OutCsta Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axPtn As Short,
      ByVal cstaMode As Short) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)	
devNo	***デバイス番号*** (0始まり)	
axPtn	***軸パターン***	
cstaMode	***モード選択***	0: LSI外部への出力なし(自LSI内の指定軸パターンのみ対象)
		1: LSI外部への信号出力もする

説明

同時スタート信号待ちとなっている軸は、この信号の入力によりスタートします。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

[VisualC++]

```
MNETM4OutCsta(0, 3, 0x3, 1);
```

[VisualBasic]

```
MNETM4OutCsta 0, 3, &H3, 1
```

[VB.NET]

```
MNETM4OutCsta(0, 3, &H3, 1)
```

ライン 0, デバイス 3 の軸パターン 3h(軸 0,1)に CSTA 信号を出力します。
(外部への出力も行なわれます)

7－1－1－28. MNETM4OutCstp

機能

同時停止(CSTP)信号出力

書式

[VisualC++]

```
short __stdcall MNETM4OutCstp(
      short lineNo,
      short devNo);
```

[VisualBasic]

```
Declare Function MNETM4OutCstp Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM4OutCstp Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short) As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）

説明

・同時停止信号が有効に設定されている軸は、この信号の入力により停止します。
・この信号の有効/無効、および入力時に即停止とするか減速停止とするかの設定は MNETM4SetPar 関数で行ないます。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照）

使用例

[VisualC++]

```
MNETM4OutCstp(0, 3);
```

[VisualBasic]

```
MNETM4OutCstp 0, 3
```

[VB.NET]

```
MNETM4OutCstp(0, 3)
```

ライン 0, デバイス 3 の CSTP 信号を出力します。(外部への出力も行なわれます)

7－1－1－29. MNETM4SetPar

機能

パラメータ設定

書式

[VisualC++]

```
short __stdcall MNETM4SetPar(
      short lineNo,
      short devNo,
      short axNo,
      short itemNo,
      long val);
```

[VisualBasic]

```
Declare Function MNETM4SetPar Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axNo As Integer,
      ByVal itemNo As Integer,
      ByVal val As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM4SetPar Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axNo As Short,
      ByVal itemNo As Short,
      ByVal val As Integer) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
itemNo	***項目選択*** (7－3－3－10. パラメータ項目番号定義参照)
val	***設定値***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

[VisualC++]

```
MNETM4SetPar(0, 3, 0, MNETM4_PAR_OUTPLS_TYPE, 0);
```

[VisualBasic]

```
MNETM4SetPar 0, 3, 0, MNETM4_PAR_OUTPLS_TYPE, 0
```

[VB.NET]

```
MNETM4SetPar(0, 3, 0, MNETM4_PAR_OUTPLS_TYPE, 0)
```

ライン 0, デバイス 3,軸 0 のパラメータ「パルスタイプ」を 0 にセットします。

7－1－1－30. MNETM4GetPar

機能

パラメータ取得

書式

[VisualC++]

```
short __stdcall MNETM4GetPar(
      short lineNo,
      short devNo,
      short axNo,
      short itemNo,
      long* val);
```

[VisualBasic]

```
Declare Function MNETM4GetPar Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axNo As Integer,
      ByVal itemNo As Integer,
      ByRef val As Long) As Integer
```

[VB.NET]

```
Declare Function MNETM4GetPar Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axNo As Short,
      ByVal itemNo As Short,
      ByRef val As Integer) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
itemNo	***項目選択*** (7－3－3－10. パラメータ項目番号定義参照)
val	***取得データ格納変数***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

[VisualC++]

```
long data;

MNETM4GetPar(0, 3, 0, MNETM4_PAR_OUTPLS_TYPE, &data);
```

[VisualBasic]

```
Dim data As Long

MNETM4GetPar 0, 3, 0, MNET_M4_PAR_OUTPLS_TYPE, data
```

[VB.NET]

```
Dim data As Integer

MNETM4GetPar(0, 3, 0, MNET_M4_PAR_OUTPLS_TYPE, data)
```

ライン0, デバイス3,軸0のパラメータ「パルスタイプ」の設定値を変数dataに取得します。

7－1－1－31. MNETM4SetDLLPar

機能

4軸モーションデバイス関連DLL内部パラメータ設定

書式

[VisualC++]

```
short __stdcall MNETM4SetDLLPar(
      short lineNo,
      short devNo,
      short axNo,
      short itemNo,
      char val);
```

[VisualBasic]

```
Declare Function MNETM4SetDLLPar Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer,
      ByVal axNo As Integer,
      ByVal itemNo As Integer,
      ByVal val As Byte) As Integer
```

[VB.NET]

```
Declare Function MNETM4SetDLLPar Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short,
      ByVal axNo As Short,
      ByVal itemNo As Short,
      ByVal val As Byte) As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
itemNo	***項目選択*** (7－3－3－11. DLL内部パラメータ項目番号定義参照)
val	***設定値***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

[VisualC++]

```
MNETM4SetDLLPar(0, 3, 0, MNETM4_DLLPAR_PRECHK_DIS, 1);
```

[VisualBasic]

```
MNETM4SetDLLPar 0, 3, 0, MNETM4_DLLPAR_PRECHK_DIS, 1
```

[VB.NET]

```
MNETM4SetDLLPar(0, 3, 0, MNETM4_DLLPAR_PRECHK_DIS, 1)
```

ライン 0, デバイス 3,軸 0 の DLL 内部パラメータ「動作前確認省略」を 1(省略)にセットします。

7－1－1－32. MNETM4Reset

機能

4軸モーションデバイスリセット

書式

[VisualC++]

```
short __stdcall MNETM4Reset(
      short lineNo,
      short devNo);
```

[VisualBasic]

```
Declare Function MNETM4Reset Lib "PPCI_L112.dll" (
      ByVal lineNo As Integer,
      ByVal devNo As Integer) As Integer
```

[VB.NET]

```
Declare Function MNETM4Reset Lib "PPCI_L112.dll" (
      ByVal lineNo As Short,
      ByVal devNo As Short) As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）

備考

本関数の実行により、対象のパルスコントロール LSI:PCL6143 のレジスタはリセットされます。
したがって対象のデバイスに設定したパラメータなどは再設定を行う必要があります。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照）

使用例

[VisualC++]

```
MNETM4Reset (0, 3);
```

[VisualBasic]

```
MNETM4Reset 0, 3
```

[VB.NET]

```
MNETM4Reset(0, 3)
```

ライン0, デバイス3の4軸モーションデバイスをリセットします。

7－1－1－33. MNETM4ReadParFile

機能

パラメータファイル読み出し
指定されたパラメータファイルを読み出し、読み出した値をレジスタに設定します。

書式

[Visual C++]

```
short __stdcall MNETM4ReadParFile(
                    short lineNo,
                    short devNo,
                    const char* filename);
```

[Visual Basic]

```
Declare Function MNETM4ReadParFile Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal filename As String)
                          As Integer
```

[VB.NET]

```
Declare Function MNETM4ReadParFile Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal filename As String)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
filename	***ファイル名***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照）

使用例

[Visual C++]

```
MNETM4ReadParFile(0, 0, “ABC.par”);
```

[Visual Basic]

```
MNETM4ReadParFile 0, 0, “ABC.par”
```

[VB.NET]

```
MNETM4ReadParFile(0, 0, “ABC.par”)
```

パラメータファイル”ABC.par”を読み出し、ライン0、デバイス0に設定します。

7－1－1－34. MNETM4WriteParFile

機能

パラメータファイル書き込み
指定されたファイルに環境設定などのレジスタデータを書き込みます。

書式

[Visual C++]

```
short __stdcall MNETM4WriteParFile(
                    short lineNo,
                    short devNo,
                    const char* filename);
```

[Visual Basic]

```
Declare Function MNETM4WriteParFile Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal filename As String)
                          As Integer
```

[VB.NET]

```
Declare Function MNETM4WriteParFile Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal filename As String)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
filename	***ファイル名***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(5－3－3. 戻り値 ならびに 7－3－4. 戻り値参照)

使用例

[Visual C++]

```
MNETM4WriteParFile(0, 0, “XYZ.txt”);
```

[Visual Basic]

```
MNETM4WriteParFile 0, 0, “XYZ.txt”
```

[VB.NET]

```
MNETM4WriteParFile(0, 0, “XYZ.txt”)
```

環境設定などのレジスタデータをファイル”XYZ.txt”に書き込みます。

7−2. 機能操作説明

7−2−1. データ取得

7−2−1−1. 状態取得

４軸モーションデバイスの入出力信号状態、ステータスビット、および軸動作状態は以下の２つの関数を使用し取得可能です。

[MNETM4GetBitData 関数]

対象ビット(データ)を指定し、その項目のデータを取得します。

記述例

[VisualC++]

```
short data;

MNETM4GetBitData(0, 3, 1, MNETM4_SEL_BIT_P0, &data);
```

[VisualBasic]

```
Dim data As Integer

MNETM4GetBitData 0, 3, 1, MNETM4_SEL_BIT_P0, data
```

[VB.NET]

```
Dim data As Short

MNETM4GetBitData(0, 3, 1, MNETM4_SEL_BIT_P0, data)
```

ライン0のデバイス3,軸1のビットP0の状態を変数dataに取得します。

[MNETM4GetBitDataStruct 関数]

ビットデータ構造体にビット(データ)を一括で取得します。

使用例

[VisualC++]

```
TMNETM4BitData bit = {0};       // ビットデータ構造体定義

MNETM4GetBitDataStruct(0, 3, 1, &bit);// ライン0のデバイス3,軸1のビットデータを一括取得
If (bit.spel != 0)              // spel (プラスEL)ビットによる判定
  m_pel.SetCheck(TRUE);         // ON時:チェックON
ELse
  m_pel.SetCheck(FALSE);        // OFF時: チェックOFF

If (bit.salm != 0)              // salm(ALM)ビットによる判定
  m_alm.SetCheck(TRUE);         // ON時: チェックON
ELse
  m_alm.SetCheck(FALSE);        // OFF時: チェックOFF
```

[VisualBasic]

```
Dim bit As TMNETM4BitData       ’ビットデータ構造体定義

MNETM4GetBitDataStruct 0, 3, 1, bit  ’ライン0のデバイス3,軸1のビットデータを一括取得
If (bit.pel <> 0) Then          ’spel (プラスEL)ビットによる判定
  ShpPEL.FillColor = &HFF&      ’ON時:明赤色
ELse
  ShpPEL.FillColor = &H80&      ’OFF時:暗赤色
End If

If (bit.alm <> 0) Then          ’salm(ALM)ビットによる判定
  ShpAlm.FillColor = &HFF&      ’ON時:明赤色
ELse
  ShpAlm.FillColor = &H80&      ’OFF時:暗赤色
End If
```

[VB.NET]

```
Dim bit As TMNETM4BitData       ’ビットデータ構造体定義

MNETM4GetBitDataStruct 0, 3, 1, bit  ’ライン0のデバイス3,軸1のビットデータを一括取得
If (bit.pel <> 0) Then          ’spel (プラスEL)ビットによる判定
  ShpPEL.BackColor = System.Drawing.Color.Red  ’ON時:明赤色
ELse
  ShpPEL.BackColor = System.Drawing.Color.DarkRed       ’OFF時:暗赤色
End If

If (bit.alm <> 0) Then          ’salm(ALM)ビットによる判定
  ShpAlm.BackColor = System.Drawing.Color.Red  ’ON時:明赤色
ELse
  ShpAlm.BackColor = System.Drawing.Color.DarkRed       ’OFF時:暗赤色
End If
```


Motionnet®
Remote I/O & Remote Motion

7−2−1−2. カウンタ取得

４軸モーションデバイスのカウンタデータの取得は、MNETM4GetCounter 関数で行います。

記述例

[VisualC++]

```
long data;

MNETM4GetCounter(0, 3, 1, MNET_SEL_COUNTER1, &data);
```

[VisualBasic]

```
Dim data As Long

MNETM4GetCounter 0, 3, 1, MNET_SEL_COUNTER1, data
```

[VB.NET]

```
Dim data As Integer

MNETM4GetCounter(0, 3, 1, MNET_SEL_COUNTER1, data)
```

ライン 0 のデバイス 3,軸 1 のカウンタ 1 のデータを変数 data に取得します。

7−2−1−3. 停止確認

停止確認を行う方法は大きく分けて以下の 2 つの方法が利用できます。

[MNETM4GetBitDataStruct 関数、MNETM4GetBitData 関数]

指定した軸の停止に関連するビットやデータなどの取得を行い判定します。

[MNETM4StopChk 関数]

関数内部で軸の停止に関連するビットやデータが取得され､軸パターンで指定した軸の停止/動作中判別、正常停止/異常停止などが判定された結果を取得できます。

7−2−1−4. エラー割り込み要因取得

エラー割り込み要因の取得は MNETM4GetREST 関数を使用します。
このデータは読出しによりリセットされます。

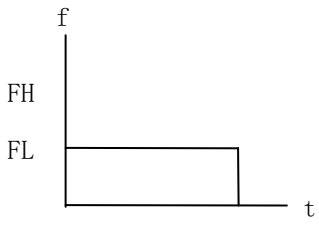
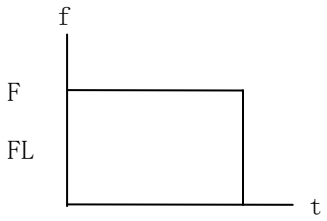
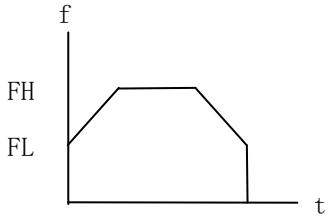
7−2−1−5. イベント割り込み要因取得

イベント割り込み要因の取得は MNETM4GetRIST 関数を使用します。
このデータは読出しによりリセットされます。 またイベント割込の設定は MNETM4SetPar 関数で行います。

7-2-2. スタートモード

スタートモードは下記 3 パターンがあります。

MNETM4SetSpeed 関数での速度データ設定時、速度データ構造体のメンバー「StartMode」に、また移動命令関数の引数に設定するデータとその識別子は次のようになります。

識別子	値	内容
MNET_STAMODE_FL	0050h	FL定速  速度データ構造体メンバー「SpdL」速度での定速動作となります。
MNET_STAMODE_FH	0051h	FH定速  速度データ構造体メンバー「SpdH」速度での定速動作となります。
MNET_STAMODE_UD	0053h	加減速  速度データ構造体メンバー「SpdL」速度から「SpdH」速度に加速、定速移動し、減速して停止します。 加速度はメンバー「Acc」、減速度は「Dec」で設定します。

Motionnet®
Remote I/O & Remote Motion

7−2−3. 速度データ構造体

MNETM4SetSpeed 関数での速度データの設定に使用します。

書式

[VisualC++]

```
struct TMNETM4Spd {
  double SpdL[4];            // FL速度
  double SpdH[4];            // FH速度
  double Acc[4];             // 加速時間
  double Dec[4];             // 減速時間
  double SAccSect[4];        // 加速S字範囲
  double SDecSect[4];        // 減速S字範囲
  short AdcMtd[4];           // 加減速指定方法 (時間/レート)
  short SMode[4];            // S字モード (0 : S字未使用, 1 : S字使用)
};
```

[VisualBasic]

```
Type TMNETM4Spd
  SpdL(0 To 3) As Double       '/ FL速度
  SpdH(0 To 3) As Double       '/ FH速度
  Acc(0 To 3) As Double        '/ 加速時間
  Dec(0 To 3) As Double        '/ 減速時間
  SAccSect(0 To 3) As Double   '/ 加速S字範囲
  SDecSect(0 To 3) As Double   '/ 減速S字範囲
  AdcMtd(0 To 3) As Integer    '/ 加減速指定方法 (時間/レート)
  SMode(0 To 3) As Integer     '/ S字モード (0 : S字未使用, 1 : S字使用)
End Type
```

[VB.NET]

```
<StructLayout(LayoutKind.Sequential)> Structure TMNETM4Spd
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim SpdL() As Double '/FL 速度
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim SpdH() As Double '/FH 速度
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim Acc() As Double '/加速時間
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim Dec() As Double '/減速時間
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim SAccSect() As Double '/加速 S 字範囲
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim SDecSect() As Double '/減速 S 字範囲
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim AdcMtd() As Short '/加減速指定方法
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim SMode() As Short '/S 字モード
  Public Sub Initialize()
    ReDim SpdL(3)
    ReDim SpdH(3)
    ReDim Acc(3)
    ReDim Dec(3)
    ReDim SAccSect(3)
    ReDim SDecSect(3)
    ReDim AdcMtd(3)
    ReDim SMode(3)
  End Sub
End Structure
```

[メンバー]

メンバー	内容	設定値	設定可能範囲
SpdL[4]	FL速度	PPS	0.3～9.8MPPS
SpdH[4]	FH速度	PPS	0.3～9.8MPPS
Acc[4]	加速データ	AdcMtdの指定に応じ,加速レートあるいは加速時間として扱われます。	0以上のデータを設定(注1)
Dec[4]	減速データ	AdcMtdの指定に応じ,減速レートあるいは減速時間として扱われます。	0以上のデータを設定(注1)
SAccSect[4]	加速S字区間	設定値0の場合、直線部分のないS字動作となります	0～(SpdH-SpdL)/2(注2)
SDecSect[4]	減速S字区間	設定値0の場合、直線部分のないS字動作となります	0～(SpdH-SpdL)/2(注2)
AdcMtd[4]	加減速度指定方法	0 : レート(1秒あたりの増分/減分パルス) 1 : 時間(秒)	左記から選択
Smode[4]	S字モード	0 : S字無し 1 : S字有り	左記から選択

(注1) AdcMtdで設定した加減速度指定方法に従い、FL速度、FH速度、S字設定、加減速データからパルスコントロールLSI:PCL6143のレジスタ設定値が計算されます。
計算された値がレジスタの最大値/最小値を超えた場合にエラーとするか、最大値/最小値に自動補正するかはパラメータで選択できます。詳細は(7－2－5－2.加減速設定の自動補正)の項を参照ください。

(注2) (7－2－6. S字動作設定)の項を参照ください。

設定されるデータ

MNETM4SetSpeed関数で設定されるデータはスタートモード別に以下のようになります。
また後述の「設定上の注意」を必ずご確認ください。

FL 定速の場合

SpdL : FL速度 必ず設定が必要です。(速度0は速度設定エラーとなります)

上記以外のメンバーのデータは無視されます。(これ以外のデータは設定されません)

FH 定速の場合

SpdL : FL速度 (速度0の場合,設定可能な最小値が設定されます)
SpdH : FH速度 必ず設定が必要です。(速度0は速度設定エラーとなります)

上記以外のメンバーのデータは無視されます。(これ以外のデータは設定されません)

加減速の場合

SpdL : FL速度 必ず設定が必要です。(速度0の場合,設定可能な最小値が設定されます)
SpdH : FH速度 必ず設定が必要です。(速度0は速度設定エラーとなります)
Acc : 加速データ 必ず設定が必要です。
Dec : 減速データ 必ず設定が必要です。
SAccSect : 加速S字範囲
SDecSect : 減速S字範囲
AdcMtd : 加減速指定方法 必ず設定が必要です。
SMode : S字使用/未使用 必ず設定が必要です。

S字範囲の設定は必須ではありません。

設定上の注意

① VisualC++での使用の場合

構造体定義時にメンバーを下記のように初期化(0クリア)してください。

```
TMNETM4Spd spd = {0};   //構造体定義と初期化(0クリア)
```

初期化を行わない場合、メンバーに残った不定データにより正しい動作が行われない可能性があります。

(VisualBasicの場合は定義時に自動的に初期化されますので必要ありません)

② VB.NETでの使用の場合

構造体定義時にメンバーを下記のように初期化してください。

```
Dim spd As New TMNETM4Spd   ’速度データ構造体定義
spd.Initialize()            ’構造体初期化
```

③ DLL内部の速度計算について

SpdL/SpdHに設定された速度は2つの速度のうち大きい速度を基準に、速度設定できる速度間隔が決まりLSIに設定される速度が演算されます。 設定される速度は演算により丸められる可能性があることをご承知おきください。詳細については(7－2－9－1. 速度倍率)の項を参照ください。

④ 動作起動関数の引数「スタートモード」について

直近に実行された MNETM4SetSpeed 関数のスタートモードにより、動作起動関数(MNETM4Ptp 他(直線補間関数を除く))実行時に指定できるスタートモードには制限があり、以下のようになっております。
このデータは DLL 内に各デバイス、各軸毎に記憶/管理されております。

MNETM4SetSpeed 関数での スタートモード指定	動作起動関数で指定可能な スタートモード
MNET_STAMODE_FL(FL 定速)	MNET_STAMODE_FL(FL 定速)
MNET_STAMODE_FH(FH 定速)	MNET_STAMODE_FL(FL 定速) MNET_STAMODE_FH(FH 定速)
MNET_STAMODE_UD(加減速)	MNET_STAMODE_FL(FL 定速) MNET_STAMODE_FH(FH 定速) MNET_STAMODE_UD(加減速)

上記以外のスタートモード指定時、あるいは速度が未設定の場合は「速度データ無効エラー」となります。なお、直線補間関数実行により当該軸の速度は未設定状態となりますので、直線補間関数以外の動作起動関数を実行する場合は改めて速度設定が必要となります。

7−2−4. 移動データ構造体

動作指令関数実行時、移動量などのデータの指定に使用します。

書式

[VisualC++]

```
struct TMNETM4Dst {
  long Pos[4];                // 目標位置/移動量
  short Direction[4];         // 動作方向
  short PreReg;               // 次動作書込
  short StartMode;            // スタートモード (FL/FH/UD)
};
```

[VisualBasic]

```
Type TMNETM4Dst
  Pos(0 To 3) As Long          '/ 目標位置/移動量
  Direction(0 To 3) As Integer '/ 動作方向
  PreReg As Integer            '/ 次動作書込
  StartMode As Integer         '/ スタートモード (FL/FH/UD)
End Type
```

[VB.NET]

```
<StructLayout(LayoutKind.Sequential)> Structure TMNETM4Dst
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim Pos() As Integer '/ 目標位置/移動量
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim Direction() As Short '/ 動作方向
  Dim PreReg As Short '/ プリレジスタ書込
  Dim StartMode As Short '/ スタートモード (FL/FH/UD)

  Public Sub Initialize()
    ReDim Pos(3)
    ReDim Direction(3)
  End Sub
End Structure
```

[メンバー]

メンバー	内容	設定値	設定可能範囲	備考
Pos[4]	移動量/目標値	パルス	-134,217,728～ +134,217,727	位置決め移動の場合に各軸の移動量/目標値を指定します。
Direction[4]	動作方向	0 : +方向 1 : -方向	左記から選択	連続移動、原点復帰移動の場合に各軸の動作方向を指定します。
PreReg	次動作書込み	0 : 通常書込み 1 : 次動作書込	左記から選択	(動作中の)次動作書込みの場合は1を設定します。
StartMode	スタートモード	50h : FL定速 51h : FH定速 53h : 加減速	左記から選択	

設定上の注意

VisualC++での使用の場合

構造体定義時にメンバーを下記のように初期化(0クリア)してください。

```
TMNETM4Dst dst = {0};   //構造体定義と初期化(0クリア)
```

初期化を行わない場合、メンバーに残った不定データにより正しい動作が行われない可能性があります。

(VisualBasicの場合は定義時に自動的に初期化されますので必要ありません)

VB.NETでの使用の場合

構造体定義時にメンバーを下記のように初期化してください。

```
Dim dst As New TMNETM4Dst    ’移動データ構造体定義
dst.Initialize()             ’構造体初期化
```

7−2−5. 加減速設定

7−2−5−1. 加減速指定方法

MNETM4SetSpeed関数での加減速データ設定時、加減速度を加速(減速)時間で指定するか、加速(減速)レート(1秒あたりの増分(減分)パルス)で指定するかを、速度データ構造体のメンバー「AdcMtd」に設定します。この指定に従い、メンバー「Acc」，「Dec」に設定された値は次のように扱われます。

識別子	値	内容
MNET_ADC_BY_RATE	0	レート指定(1秒あたりの増分(減分)パルス)
MNET_ADC_BY_TIME	1	時間指定　(秒)

7−2−5−2. 加減速設定の自動補正

設定された加減速データは DLL 内部で計算され、パルスコントロール LSI ： PCL6143 のレジスタに設定されますが、計算された結果がレジスタ長を超える場合や設定可能最小値: 1 に満たない場合に設定可能最大値/最小値に自動補正する機能があります。この機能はデフォルトで有効ですが、無効にしたい場合は MNETM4SetDLLPar 関数を使用し設定できます。

識別子	値	内容	設定 範囲	設定値／取得値 （網掛け：デフォルト）
MNETM4_DLLPAR_ADCCR_DIS	4E1h	加減速設定自動補正無効	0/1	0:補正する,1:補正しない

［補正する場合］

最大/最小値を超える場合は最大/最小値に補正されます。

［補正しない場合］

最大/最小値を超える場合は加速レート計算結果エラー/減速レート計算結果エラーになります。

7−2−6. S 字動作設定

S 字動作設定は MNETM4SetSpeed 関数での速度データ設定時に速度データ構造体メンバー「SMode」の設定で行います。

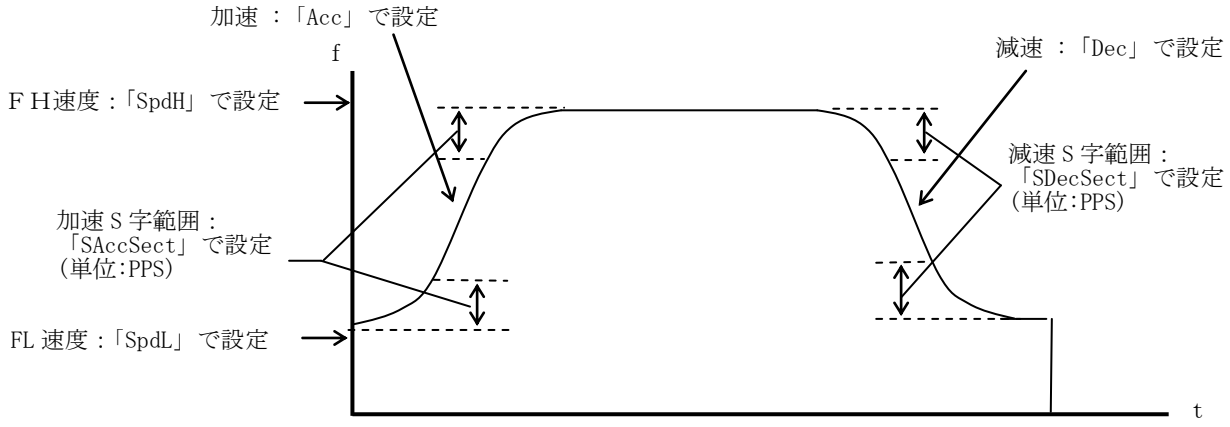
S 字動作の有無はメンバー「SMode」で指定します。0 が S 字無し、1 が S 字有りとなります。

加減時における S 字動作の範囲はメンバー「SAccSect」に指定します。

減速時における S 字動作の範囲はメンバー「SDecSect」に指定します。

S 字有り（SMode：1）で加速 S 字範囲(「SAccSect」)に 0 を設定した場合、自動的に(SpdH-SpdL)/2 の値が適用され、加速全域にわたって(直線加速部分のない)S 字動作となります。

同様に減速Ｓ字範囲(「SDecSect」)に 0 を設定した場合、自動的に(SpdH-SpdL)/2 の値が適用され、減速全域にわたって(直線減速部分のない)S 字動作となります。



加減速動作と、対応する速度データ構造体メンバーの設定を下表にまとめます。

加減速動作	「Smode」設定	「SAccSect」，「SDecSect」設定
直線加減速(全域が直線)	0	直線加減速となるため設定は無効です。
S字加減速(全域がS字)	1	0
直線部分のあるS字加減速	1	1以上、 (SpdH-SpdL)/2 以下の範囲で設定します。(注)

(注) (SpdH-SpdL)/2 を超えた場合は(SpdH-SpdL)/2 に自動補正されます。

7−2−7. 原点復帰モード

原点復帰動作の起動はMNETM4Org関数で行います。
原点復帰動作の設定はMNETM4SetPar関数で行います。

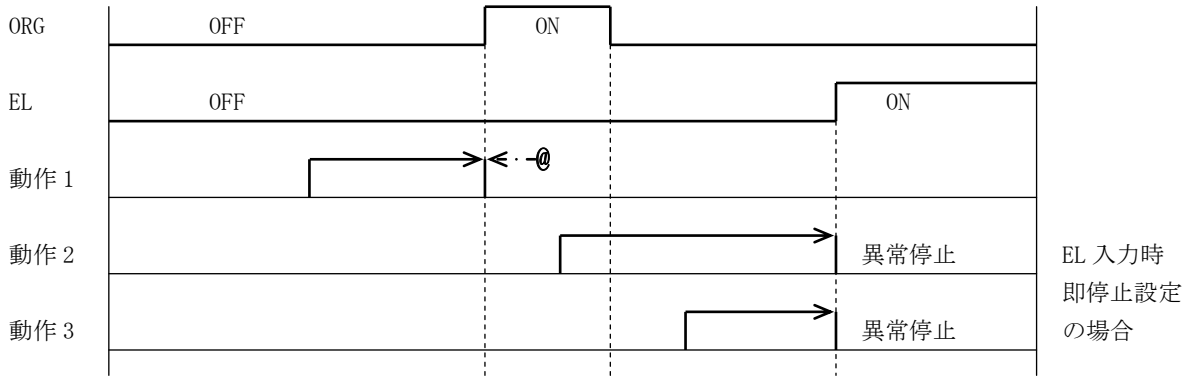
原点復帰動作における注意など

① ORG信号を使用する原点復帰動作はORG入力のOFF→ONにより処理されます。
既にORG入力がONしている場合、あるいはORG信号センサーを越えた位置にいる場合、動作方向のELまで動作しつづけることになりますのでご注意ください。

② 原点復帰動作によりCOUNTER1がリセットされます。
スタートモード/設定によりカウンタのリセット位置が異なりますのでご注意ください。

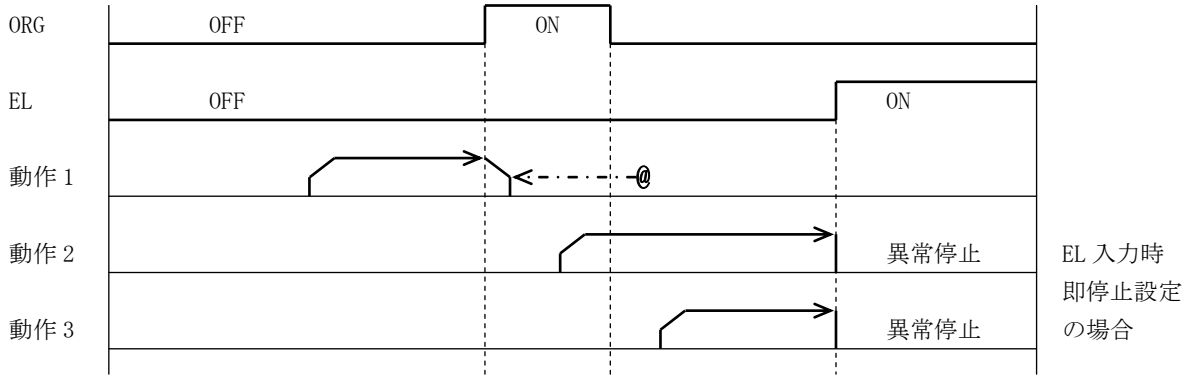
7−2−7−1. 原点復帰モード：0

□定速動作 ＜センサ：EL, ORG＞



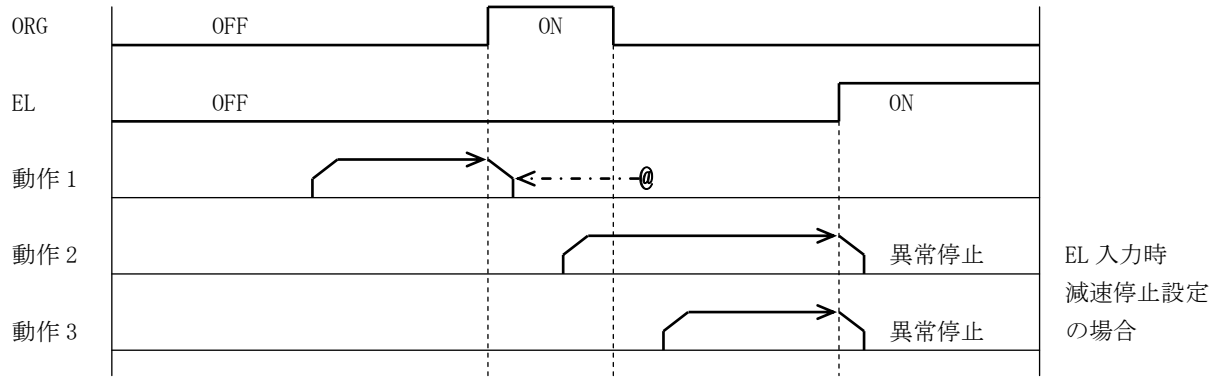
■加減速動作 ＜センサ：EL, ORG＞

正常停止時でも原点位置ではありませんが、COUNTER2(機械位置)は信頼できる値となります。

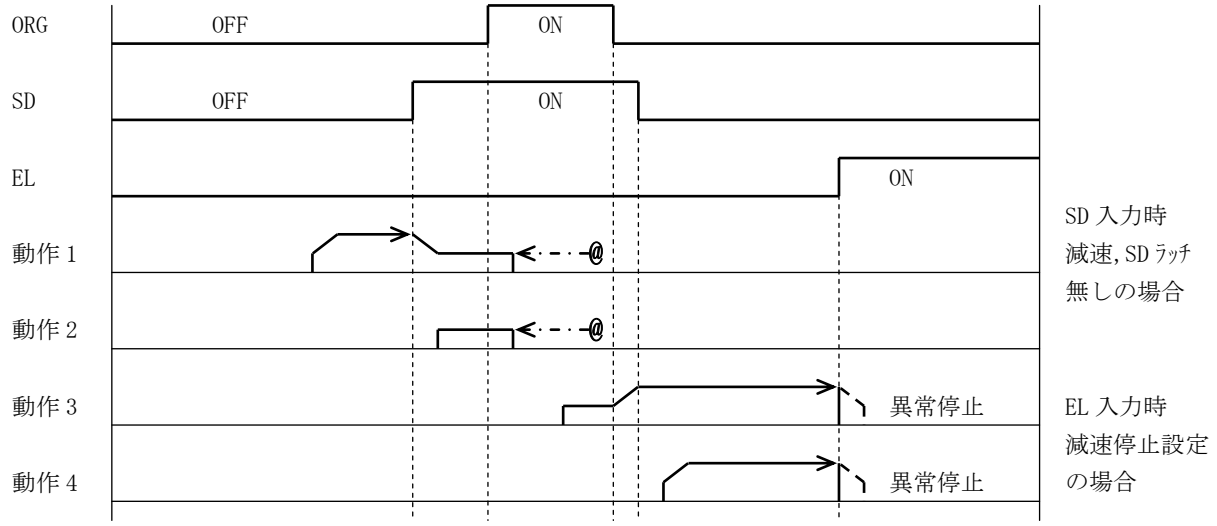


■加減速動作　＜センサ：EL, ORG＞

正常停止時でも原点位置ではありませんが、COUNTER2(機械位置)は信頼できる値となります。



■加減速動作　＜センサ：EL, SD, ORG＞

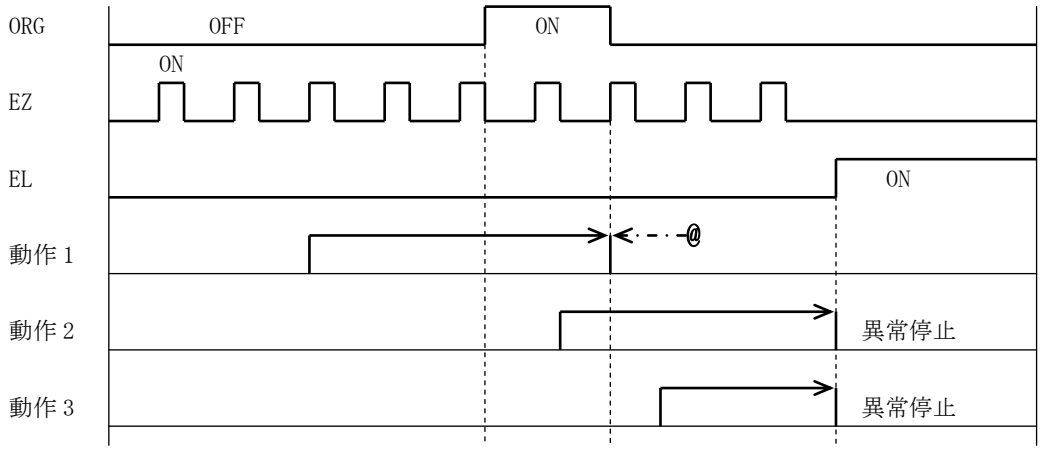


注意. @位置は原点復帰停止時に"ERC 信号を自動出力"に設定した時の ERC 信号出力のタイミングです。

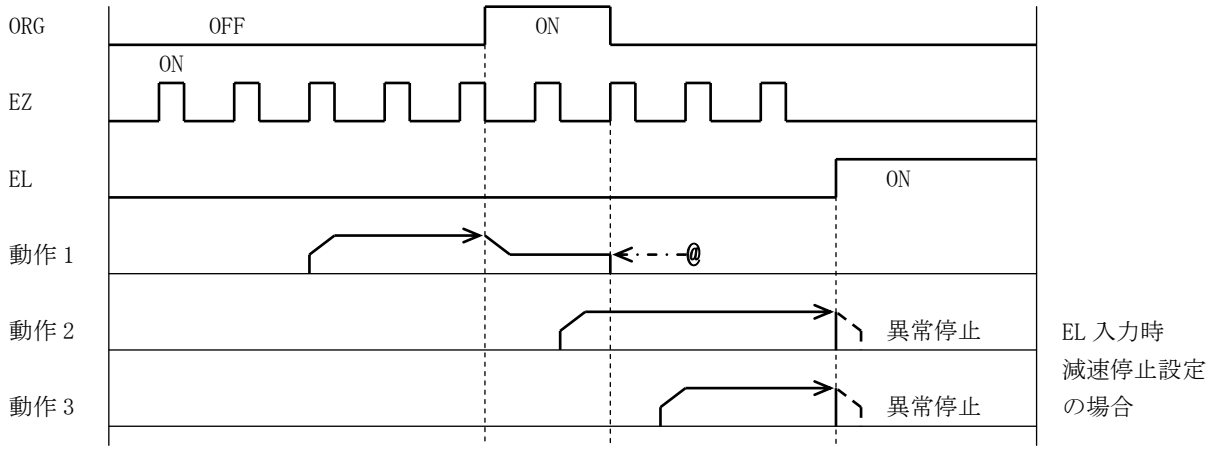
Motionnet®
Remote I/O & Remote Motion

7−2−7−2. 原点復帰動作モード：1

□定速動作　＜センサ：EL, ORG, EZ(Z 相回数 2 回目設定)＞



■加減速動作　＜センサ：EL, ORG, EZ(Z 相回数 2 回目設定)＞



注意.@位置は原点復帰停止時に"ERC 信号を自動出力"に設定した時の ERC 信号出力のタイミングです。

7−2−8. 動作前確認

本 DLL ではモーション関数実行前の状態確認を行うか/省略するかはパラメータになっており、MNETM4SetDLLPar 関数を使用し設定できます。

識別子	値	内容	設定範囲	設定値／取得値（網掛け：デフォルト）
MNETM4_DLLPAR_PRECHK_DIS	4E0h	軸動作前確認省略	0/1	0:省略しない,1:省略

［省略しない場合］

「ライン番号」、「デバイス番号」、「軸番号/軸パターン」、「デバイスが４軸モーションデバイスかどうか」、「動作中でないこと」、「動作方向の EL 入力」、「ALM 入力」、「EMG 入力」、「目的の信号入力状態(ORG,EL など)」、「次動作書込みの場合、書込み可能かどうか」の確認が行われます。

ただし実行前に上記データ確認のための数回のデータ通信が発生します。

［省略する場合］

「ライン番号」、「デバイス番号」、「軸番号/軸パターン」、「デバイスがモーションデバイスかどうか」、「動作中でないこと」、「次動作書込みの場合、書込み可能かどうか」の確認が行われますが、その他の入出力信号や指令値の確認などは行われません。

ALM などの入力がある場合、動作指令は発行後エラーとなり、エラー割り込みありビット(SERR)が ON することになります。エラー割り込み要因は MNETM4GetREST 関数で確認してください。

Motionnet®
Remote I/O & Remote Motion

7−2−9. 速度設定に関する詳細事項

7−2−9−1. 速度倍率

FL/FH 速度設定範囲は 0.3〜9.9MPPS となっておりますが、DLL 内部においては指定値に基づき計算を行い、パルスコントロール LSI ： PCL6143 の「FL/FH 速度レジスタ」と「速度倍率レジスタ」にデータを設定しております。レジスタ設定値と出力パルス速度の関係は以下のようになっております。

「出力パルス速度」 = 「FL/FH 速度レジスタ設定値」×「速度倍率」

例を挙げると速度倍率が 10 倍、出力パルス速度が 150,000PPS である場合

150,000(出力パルス速度) = 15,000(速度レジスタ設定値) × 10 倍(速度倍率)

となります。

この倍率という考え方により速度レジスタのビット長の制限を越えた速度設定を行えるようになっています。具体的には各速度設定レジスタ設定範囲は 1〜16,383 であり、それに 0.3〜610.3 の倍率を乗ずることで 0.3〜9.9MPPS という速度設定範囲を実現しています。

各速度倍率とその出力パルス速度の範囲を下表に示します。

倍率	出力速度範囲(PPS)	倍率	出力速度範囲(PPS)
0.3	0.3 ～ 4,914.9	20	20.0 ～ 327,848.8
0.5	0.5 ～ 8,192.8	50.8	50.8 ～ 833,282.4
1	1 ～ 16,392.4	101.7	101.7 ～ 1,666,564.9
2	2 ～ 32,784.8	203.4	203.4 ～ 3,333,129.8
5	5 ～ 81,962.2	406.9	406.9 ～ 6,666,259.7
10	10 ～ 163,924.4	610.3	610.3 ～ 9,999,389.6

倍率はパルス出力中に可変するものではなく、動作スタート前に設定された値に固定となります。

それぞれの倍率は表の通りの範囲を持ち、一つの動作の中では範囲を超えることはできません。

本 DLL では MNETM4SetSpeed 関数で速度データが設定されるたび、速度データ構造体メンバー「SpdL」,「SpdH」に設定されたデータの中で、大きい方の値を速度倍率選択速度とし、その速度倍率選択速度を満たす最も小さい速度倍率が自動的に選択され、各速度レジスタの値が計算/設定されています。

(小さい速度倍率のほうが設定できる速度の速度間隔が細かくなるため)

基本的には自動的に選択される速度倍率は意識する必要はありませんが、以下の場合に注意が必要です。

設定速度データの丸めについて

各速度レジスタへの設定値は整数値です。

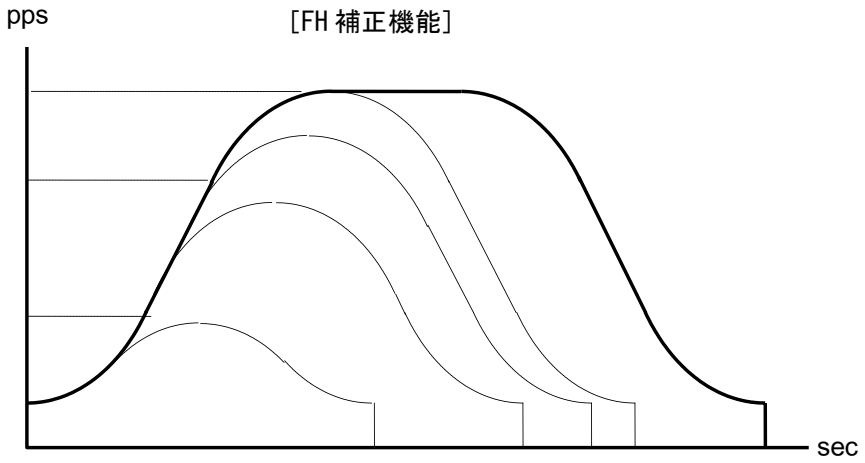
(レジスタ設定値(整数値) = 各 SpdL/SpdH 指定値(実数値) ÷ 速度倍率(実数値))

したがって SpdL/SpdH に指定された値は丸められる場合があります。

また SpdL/SpdH に指定した値が 0 に丸められた場合、設定速度は選択された倍率の最小値に自動的に補正されます。

7−2−9−2. FH 補正

FH 補正とは MNETM4Ptp 関数などの位置決め動作が加減速で行なわれる際に、移動量が少なく設定した FH 速度に到達できず三角駆動となる場合、自動的に FH 速度を低下させて三角駆動を回避する機能です。



移動量による最高速度の自動修正

FH 補正機能の有効/無効は MNETM4SetPar 関数の項目「FH 補正有効/無効」で設定します。

識別子	値	内容	設定範囲	設定値／取得値 （網掛け：デフォルト）
MNETM4_PAR_FH_CORRECT_DIS	4F1h	FH補正有効/無効	0/1	0:有効, 1:無効

注意

加速と減速の傾きや S 字範囲設定が異なる速度設定の場合、本パラメータの設定にかかわらず、FH 補正が行われます。

7−2−10. SD入力

SD 信号の入力により FH 速度で動作中の軸を FL 速度に減速することができます。
SD 信号入力による減速を行う場合はあらかじめ MNETM4SetSpeed 関数でスタートモードに「加減速」を指定し FL 速度および加減速データの設定を行っておく必要があります。
(スタートモード「FL 定速」,「FH 定速」で速度データ設定を行った場合、加減速データは設定されません。
(7-2-3. 速度データ構造体)-「設定されるデータ」を参照ください。)

SD 信号入力の有効/無効、信号論理、他のパラメータは MNETM4SetPar 関数で設定します。

SD 入力処理に関連するパラメータは以下の通りです。

識別子	値	内容	設定範囲	設定値／取得値（網掛け：デフォルト）
MNETM4_PAR_SD_ENA	4F7h	SD有効(注1)	0/1	0:無効, 1:有効
MNETM4_PAR_SDLTC_ENA	411h	SDラッチ有効(注2)	0/1	0:無効, 1:有効
MNETM4_PAR_SD_LOGIC	412h	SD信号論理	0/1	0:負論理, 1:正論理

(注 1)SD 有効/無効の設定は各動作起動時に設定されます。動作中の移動には設定は適用されません。
(注 2)SD ラッチが無効である場合、SD 入力中のみ FL 速度までスローダウンし、SD 入力が OFF すると、FH 速度まで再加速します。SD ラッチが有効である場合、動作停止まで SD 入力はラッチされ SD 入力が OFF しても再加速しません。

7−2−11. PCS入力

PCS 信号の入力により PCS 信号が ON した位置を基準にした位置決めができます。
PCS 信号入力の有効/無効、信号論理、他のパラメータは MNETM4SetPar 関数で設定します。

PCS 入力処理に関連するパラメータは以下の通りです。

識別子	値	内容	設定範囲	設定値／取得値（網掛け：デフォルト）
MNETM4_PAR_PCS_ENA	4F6h	PCS信号有効(注1)	0/1	0:無効, 1:有効
MNETM4_PAR_PCS_LOGIC	41Eh	PCS信号論理	0/1	0:負論理, 1:正論理

(注 1)PCS 有効/無効の設定は各動作起動時に設定されます。動作中の移動には設定は適用されません。

PCS 動作を行う手順

(MNETM4SetPar 関数で PCS 信号を「有効」に設定します。)
① MNETM4PtpInc 関数に PCS 入力 ON からの位置決め量をセットし実行します。
② 指定した速度で軸が動作し、PCS 入力が ON になった地点から位置決めが行われます。
(MNETM4SetPar 関数で PCS 信号を「無効」に設定します。)

また MNETM4SoftPCS 関数を使用することで、PCS 入力を代行するコマンドを発行できます。

注意

PCS 動作を行なうことのできる動作起動関数は MNETM4PtpInc のみとなります。また PCS 信号有効の場合、次動作書込みは行えません。

7−2−12. 同期スタート

同時スタートとは、軸をスタート待ち状態にし、同時スタート(CSTA)信号入力や、設定した(LSI)内部同期信号入力、あるいは指定した軸の停止などによりスタートさせる動作です。

同期スタートに関連するパラメータは MNETM4SetPar 関数で設定します。

同期スタートに関連するパラメータは以下の通りです。

識別子	値	内容	設定範囲	設定値／取得値（網掛け：デフォルト）
MNETM4_PAR_MSY_MODE	4F9h	同期スタートモード選択(注1)(注2)	0～3	0:無効 1:同時スタート(CSTA)信号待ち 2:内部同期スタート信号待ち 3:指定軸の停止待ち
MNETM4_PAR_CSTA_TYPE	481h	CSTA入力仕様	0/1	0:レベルトリガ, 1:エッジトリガ
MNETM4_PAR_SYNC_OUT_MODE	485h	内部同期信号出力タイミング選択(注2)	0～15	1:コンパレータ1条件成立時 2:コンパレータ2条件成立時 8:加速開始時 9:加速終了時 10:減速開始時 11:減速終了時 その他:内部同期信号出力OFF
MNETM4_PAR_SYNC_INP	486h	内部同期信号スタート時入力選択(注2)	0～3	0:軸0が出力した内部同期信号 1:軸1が出力した内部同期信号 2:軸2が出力した内部同期信号 3:軸3が出力した内部同期信号
MNETM4_PAR_MSY_SAX	4FAh	他軸停止スタート設定(注1)(注2)	0～15	停止確認する軸パターンを4ビットで 設定します。 ビット0:軸0の停止を確認します ビット1:軸1の停止を確認します ビット2:軸2の停止を確認します ビット3:軸3の停止を確認します デフォルト:0h

(注 1)これらの設定は各動作起動時に設定されます。動作中の移動には設定は適用されません。

(注2) 直線補間指令時の設定については以下の制限があります。

①「同期スタートモード選択」設定は補間を行なう各軸で同一である必要があります。

②「同期スタートモード選択」が「内部同期スタート信号待ち」の場合は「内部同期信号スタート時入力選択」設定も補間を行なう各軸で同一である必要があります。

③ 「同期スタートモード選択」が「指定軸の停止待ち」の場合は「他軸停止スタート設定」も補間を行なう各軸で同一である必要があります。

同時スタート入力(CSTA)による動作を行う手順

(MNETM4SetPar 関数で「同期スタートモード選択」を「同時スタート(CSTA)信号待ち」に設定します。)

① CSTA設定が「有効」の軸は移動命令を実行すると、同時スタート(CSTA)信号入力待ち状態となります。(軸の状態はMNETM4GetBitDataStruct関数などで取得可能です)

② 同時スタート(CSTA)信号の入力により、動作が開始されます。
(MNETM4SetPar 関数で「同期スタートモード選択」を「無効」に設定します。)

同時スタート待ち状態の解除

同時スタート待ち状態の解除はMNETM4Stop関数を停止モード：「即停止」で実行することで行われます。

7−2−13. 同時停止

同時停止は、同時停止(CSTP)信号の入力により複数軸を同時停止させる機能です。
CSTP信号の有効無効および、信号論理などのパラメータはMNETM4SetPar関数で設定します。

CSTP 入力処理に関連するパラメータは以下の通りです。

識別子	値	内容	設定範囲	設定値／取得値 （網掛け：デフォルト）
MNETM4_PAR_CSTP_ENA	4F2h	CSTP有効	0/1	0:無効, 1:有効
MNETM4_PAR_AUTO_CSTP_ENA	4F3h	CSTP自動出力有効	0/1	0:無効, 1:有効 （異常停止時にCSTP自動出力）
MNETM4_PAR_CSTP_PROC	484h	CSTP入力時処理	0/1	0:即停止, 1:減速停止

同時ストップ動作を行う手順

(MNETM4SetPar 関数で CSTP を「有効」に設定します。)
CSTP設定が「有効」の軸は、動作中に同時ストップ(CSTP)信号入力がONした場合、設定した入力時処理で停止します。
(MNETM4SetPar 関数で CSTA を「無効」に設定します。)

注意

直線補間動作時は上記の設定にかかわらず、CSTA 有効, CSTP 自動出力有効となります。

7−2−14. 基本固定パラメータ

本 DLL で４軸モーションデバイスを操作する場合に、設定が必要な基本固定パラメータの設定内容は以下の通りです。モーションデバイスのリセット後、モーションデバイス複合関数ご利用前に必ずこの基本固定パラメータを設定ください。

識別子	値	内容	設定範囲	設定値
MNETM4_PAR_BASE_DATA	0h	基本固定パラメータ設定	0	0を設定ください

設定される内容

① INT 出力マスク
② 汎用入出力設定 P0,P1:出力
③ 原点復帰完了でカウンタ 1 リセット
④ 入力フィルタ、パルサーフィルタ、エンコーダフィルタ有効

7−2−15. コンパレータ

パルスコントロール LSI:PCL6143 は 28 ビットのコンパレータを 2 つ内蔵しており、カウンタ値との比較を行うことができます。コンパレータ 1 はカウンタ 1 との比較、コンパレータ 2 はカウンタ 2 との比較となります。

コンパレータを利用し、以下の動作(処理)などを行なうことができます。

◆比較結果の外部出力(CP1)
◆比較結果を示すステータス(Scp1,Scp2)の利用
◆内部同期信号を利用しての同期スタート動作

これらコンパレータの動作仕様は MNETM4SetPar 関数で設定します。

識別子	値	内容	設定範囲	設定値／取得値 (網掛け：デフォルト)
MNETM4_PAR_CMP1_MODE	461h	コンパレータ1比較方法(成立条件)	0～3	0:コンパレータ機能OFF 1:コンパレータ1データ = カウンタ1 2:コンパレータ1データ > カウンタ1 3:コンパレータ1データ < カウンタ1
MNETM4_PAR_CMP1_DATA	463h	コンパレータ1データ	右記	-134,217,728～134,217,727 (8000000h～7ffffffh) 0
MNETM4_PAR_CMP2_MODE	465h	コンパレータ2比較方法(成立条件)	0～3	0:コンパレータ機能OFF 1:コンパレータ2データ = カウンタ2 2:コンパレータ2データ > カウンタ2 3:コンパレータ2データ < カウンタ2
MNETM4_PAR_CMP2_DATA	467h	コンパレータ2データ	右記	-134,217,728～134,217,727 (8000000h～7ffffffh) 0

Motionnet®
Remote I/O & Remote Motion

7−2−16. イベント割り込み設定

イベント割り込み要因の設定は MNETM4SetPar 関数で行います。
イベント割り込みを発生させたい内容に対応するビットを１にしたデータを設定ください。

Bit	ﾋﾞｯﾄ名	内容
0	IREN	正常停止時
1	IRNM	プリレジスタ書込可能時
2	IRUS	加速開始時
3	IRUE	加速終了時
4	IRDS	減速開始時
5	IRDE	減速終了時
6	IRC1	コンパレータ 1 条件成立時
7	IRC2	コンパレータ 2 条件成立時
8	IRLT	LTC 入力でのﾗｯﾁ時(ﾊﾟﾗﾒｰﾀ「LTC1/2 ﾊｰﾄﾞ入力無効」が共に 1 の場合は動作しません)
9	IROL	ORG 入力 ON 時
10	IRSD	SD 入力 ON 時
11	IRDR	DR 入力変化時
12	IRSA	CSTA 入力 ON 時
31-13	未定義	（常に 0 を設定して下さい。）

イベント割り込み発生の有無は MNETM4GetBitData 関数または MNETM4GetBitDataStruct 関数で確認できます。イベント割り込み要因の読出しは MNETM4GetRIST 関数で行います。

7−2−17. ラッチ

カウンタ 1,2 は下記の方法で値をラッチでき、ラッチした値の読出しは MNETM4GetLtc 関数で行えます。

①LTC 信号 OFF→ON 時
②MNETM4SoftLtc 関数の実行(ラッチコマンドの書込み)

ラッチ動作の設定/処理などは MNETM4SetPar 関数で設定します。

ラッチに関連するパラメータは以下の通りです。

識別子	値	内容	設定範囲	設定値／取得値（網掛け：デフォルト）
MNETM4_PAR_LTC_LOGIC	41Fh	LTC信号論理	0/1	0:立下り, 1:立上り
MNETM4_PAR_LTC1_INP_DIS	420h	LTC入力によるカウンタ1のラッチ無効	0/1	0:有効 ,1:無効
MNETM4_PAR_LTC2_INP_DIS	421h	LTC入力によるカウンタ2のラッチ無効	0/1	0:無効, 1:有効

7−2−18. 直線補間

MNETM4Line 関数、MNETM4LineInc 関数の利用により直線補間動作を行なうことができます。

直線補間動作の動作速度は全補間軸間のベクトル速度(合成速度)となります。
動作速度の設定は軸パターンに関わらず、速度データ構造体の軸 0 のデータに設定します。

直線補間を行なうにあたり、以下の注意点があります。
(1)直線補間動作関数の実行によりMNETM4SetSpeed関数で設定した、非補間動作用の速度データは未設定状態となります。
(2)直線補間時は、補間に含まれる軸の以下の設定が強制的に有効となります。
* 同時停止(CSTP)信号による停止。
* 異常停止時、同時停止(CSTP)信号自動出力
* 同時減速(CSD)信号による減速
* 減速時、FL定速動作時に同時減速(CSD)信号自動出力
(3)同期スタートを行なう場合、補間に含まれる軸の同期スタート設定を同一にしておく必要があります。
(4)軸パターンに含まれる軸が 1 軸だけの場合、補間軸全てが移動量 0 の場合はエラーとなります。
(5)次動作設定で現動作に連続して直線補間動作を行わせる際に、対象軸の現動作が同時終了とならない Ptp 動作などの場合、補間動作にはならず現動作完了により各軸がバラバラにスタートすることになってしまいます。同時にスタートさせるには現動作を補間動作とするか、「同期スタート(他軸の停止によるスタート)」をご利用ください。

7−2−19. 次動作設定

移動データ構造体のメンバー「PreReg」に 1 を設定することにより次動作の設定を行うことができます。次動作を設定した場合、現動作完了後、設定した次動作が自動的にスタートします。
次動作を設定するプリレジスタは各軸毎に 1 段用意されております。
既にプリレジスタに次動作が設定されている場合は、プリレジスタは満杯となり、次々動作は設定できません。現動作が完了することで次動作がスタートしプリレジスタが空きとなり次第、次動作が設定できる状態となります。

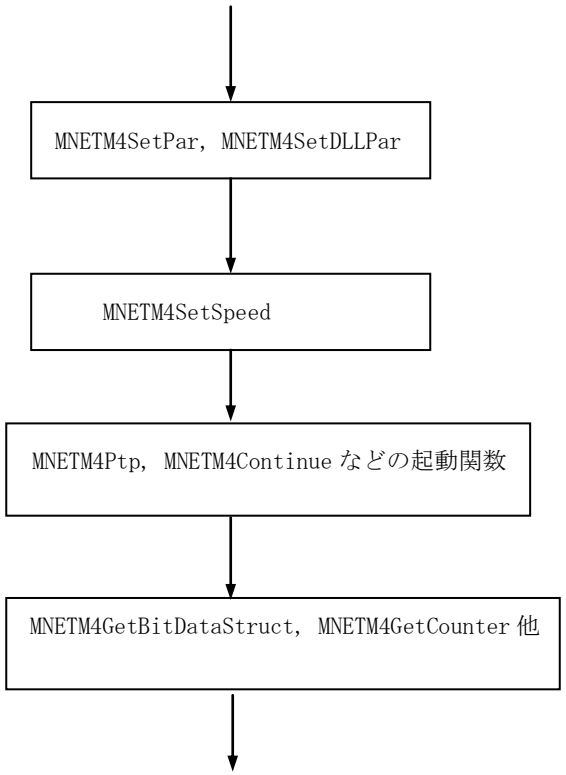
プリレジスタに設定された次動作のキャンセルは以下により行なわれます。
(1)プリレジスタに設定された次動作は、MNETM4Stop 関数または MNETM4CancelPreReg 関数を実行することでキャンセルされます。
(2)プリレジスタに設定された次動作は、ALM,±EL,EMG,CSTP 入力などによるエラー発生による停止によりキャンセルされます。

プリレジスタ利用での動作継続には以下の制限があります。
(1)連続移動、原点復帰移動などが動作中の場合、次動作に位置決め動作は設定できません。
(2)現動作中の次動作設定時にエラーが発生した場合は次動作設定は行なえません。
(3)PCS 有効設定では次動作設定は行なえません。
(4)エラー割込み要因にデータが残っている場合、「PreReg」設定’1’での動作は受け付けられません。

7－2－20. 操作の流れ

システム起動から４軸モーションデバイスを操作する流れの概略は次のようになります。

ボードのオープン，システム通信，I/O 通信開始



ローカルデバイスのパラメータデータを設定します。
（信号仕様、信号論理、動作仕様などの環境設定）

速度データの設定。
（速度、加減速度、S 字動作設定など）

動作の起動。
（移動量、動作方向などを設定し動作を起動します）

モーションデバイスの状態取得。
動作状態，信号状態，カウンタなどのモニタ

7−2−20−1. プログラム概略

Motionnet の４軸モーションデバイスを操作するプログラムコードの概略を示します。
以下のコードはあくまで流れをつかんでもらうためのものであり、関数の実行結果の取得などをはじめとする実際の動作において必要な処理の多くを省略してありますのでご注意ください。

[VisualC++]

```
short lineNo = 0;          // ライン番号
short devNo = 0;           // デバイス番号
short axNo = 0;            // 軸番号
TMNETM4Spd spd = {0};      // 速度データ構造体
TMNETM4Dst dst = {0};      // 移動データ構造体
TMNETM4BitData bit = {0}; // ビットデータ構造体
short axPtn;               // 軸パターン(軸 3〜0 を 4 ビットで表現したデータ)
short stopSts;             // 位置決め完了確認用
bool posEnd = false;       // 位置決め完了状態

PPCI_L112Open();                    // ボードのオープン
MNETCenterReset(lineNo);            // センターデバイスリセット
MNETSysComm(lineNo);                // システム通信
MNETIOCommStart(lineNo);            // I/O 通信スタート

// パラメータ設定(下記はその一部です。サンプルプログラムを参照ください)
MNETM4SetPar(lineNo, devNo, axNo, MNETM4_PAR_BASEDATA, 0);       // 共通基本データ
MNETM4SetPar(lineNo, devNo, axNo, MNETM4_PAR_OUTPLS_TYPE, 0);    // 出力パルス仕様(設定:0)
MNETM4SetPar(lineNo, devNo, axNo, MNETM4_PAR_ORG_LOGIC, 0);      // ORG 論理(負論理)
MNETM4SetPar(lineNo, devNo, axNo, MNETM4_PAR_ALM_LOGIC, 1);      // ALM 論理(正論理)

// 速度データ構造体にデータ設定
spd.SpdL[axNo] = 500;                       // FL 速度:500PPS
spd.SpdH[axNo] = 3000;                      // FH 速度:3000PPS
spd.AdcMtd[axNo] = MNET_ADC_BY_TIME;        // 加減速データ指定方法:時間
spd.Acc[axNo] = 0.5;                        // 加速時間:0.5Sec
spd.Dec[axNo] = 0.5;                        // 減速時間:0.5Sec
spd.SMode[axNo] = MNET_NON_S;               // S 字未使用
MNETM4SetSpeedNo(lineNo, devNo, axNo, &spd, MNET_STAMODE_UD); // 速度データ設定

// 移動データ構造体にデータ設定
dst.Pos[axNo] = 10000;                      // 移動量:10000 パルス
dst.StartMode = MNET_STAMODE_UD;            // スタートモード:加減速
MNETM4PtpIncNo(lineNo, devNo, axNo, &dst); // PtpInc 移動

// 位置決め完了を待つ
axPtn = (0x1 << axNo);  // 位置決め完了待ちをする軸パターンデータ作成
while (posEnd == false) {
MNETM4StopChk(lineNo, devNo, axPtn, &stopSts);  // 動作停止確認
  if (stopSts != 0) posEnd= true;
}
MNETCenterReset(lineNo);                    // センターデバイスリセット
PPCI_L112Close();                           // ボードのクローズ
```

[VisualBasic]

```
    Dim lineNo As Integer       ’ライン番号
    Dim devNo As Integer        ’デバイス番号
    Dim axNo As Integer         ’軸番号
    Dim spd As TMNETM4Spd       ’速度データ構造体
    Dim dst As TMNETM4Dst       ’移動データ構造体
    Dim bit As TMNETM4BitData   ’ビットデータ構造体
    Dim axPtn As Integer        ’軸パターン(軸3～0を4ビットで表現したデータ)
    Dim stopSts As Integer      ’位置決め完了確認用
    Dim posEnd As Boolean       ’位置決め完了状態

    PPCI_L112Open                   ’ボードのオープン
    lineNo = 0
    devNo = 0
    axNo = 0
    MNETCenterReset lineNo          ’センターデバイスリセット
    MNETSysComm lineNo              ’システム通信
    MNETIOCommStart lineNo          ’I/O通信スタート

  ’パラメータ設定(下記はその一部です。サンプルプログラムを参照ください)
    MNETM4SetPar lineNo, devNo, axNo, MNETM4_PAR_BASEDATA, 0        ’共通基本データ
    MNETM4SetPar lineNo, devNo, axNo, MNETM4_PAR_OUTPLS_TYPE, 0     ’出力パルス仕様(設定:0)
    MNETM4SetPar lineNo, devNo, axNo, MNETM4_PAR_ORG_LOGIC, 0       ’ORG論理(負論理)
    MNETM4SetPar lineNo, devNo, axNo, MNETM4_PAR_ALM_LOGIC, 1       ’ALM論理(正論理)
  ’速度データ構造体にデータ設定
    spd.SpdL(axNo) = 500                    ’FL速度:500PPS
    spd.SpdH(axNo) = 3000                   ’FH速度:3000PPS
    spd.AdcMtd(axNo) = MNET_ADC_BY_TIME     ’加減速データ指定方法:時間
    spd.Acc(axNo) = 0.5                     ’加速時間:0.5Sec
    spd.Dec(axNo) = 0.5                     ’減速時間:0.5Sec
    spd.SMode(axNo) = MNET_NON_S            ’S字未使用
    MNETM4SetSpeedNo lineNo, devNo, axNo, spd, MNET_STAMODE_UD  ’速度データ設定

    ’移動データ構造体にデータ設定
    dst.Pos(axNo) = 10000                   ’移動量:10000パルス
    dst.StartMode = MNET_STAMODE_UD         ’スタートモード:加減速
    MNETM4PtpIncNo lineNo, devNo, axNo, dst  ’PtpInc移動
    ’位置決め完了を待つ
    axPtn = &H1         ‘位置決め完了待ちをする軸パターンデータ設定
    Do While (posEnd = False)
    MNETM4StopChk lineNo, devNo, axPtn, stopSts   ’動作停止確認
       If (stopSts<>0) Then
        posEnd = True
      EndIf
    Loop

    MNETCenterReset lineNo                  ’センターデバイスリセット
    PPCI_L112Close                          ’ボードのクローズ
```

[VB.NET]

```
  Dim lineNo As Short          ’ライン番号
  Dim devNo As Short           ’デバイス番号
  Dim axNo As Short            ’軸番号
  Dim spd As New TMNETM4Spd    ’速度データ構造体
  Dim dst As New TMNETM4Dst    ’移動データ構造体
  Dim bit As TMNETM4BitData    ’ビットデータ構造体
  Dim axPtn As Short           ’軸パターン(軸3～0を4ビットで表現したデータ)
  Dim stopSts As Short         ’位置決め完了確認用
  Dim posEnd As Boolean        ’位置決め完了状態
  spd.Initialize()             ’構造体初期化
  dst.Initialize()             ’構造体初期化

  PPCI_L112Open()                    ’ボードのオープン
  lineNo = 0
  devNo = 0
  axNo = 0
  MNETCenterReset(lineNo)            ’センターデバイスリセット
  MNETSysComm(lineNo)                ’システム通信
  MNETIOCommStart(lineNo)            ’I/O通信スタート

’パラメータ設定(下記はその一部です。サンプルプログラムを参照ください)
  MNETM4SetPar(lineNo, devNo, axNo, MNETM4_PAR_BASEDATA, 0)        ’共通基本データ
  MNETM4SetPar(lineNo, devNo, axNo, MNETM4_PAR_OUTPLS_TYPE, 0)     ’出力パルス仕様(設定:0)
  MNETM4SetPar(lineNo, devNo, axNo, MNETM4_PAR_ORG_LOGIC, 0)       ’ORG論理(負論理)
  MNETM4SetPar(lineNo, devNo, axNo, MNETM4_PAR_ALM_LOGIC, 1)       ’ALM論理(正論理)

’速度データ構造体にデータ設定
  spd.SpdL(axNo) = 500                    ’FL速度:500PPS
  spd.SpdH(axNo) = 3000                   ’FH速度:3000PPS
  spd.AdcMtd(axNo) = MNET_ADC_BY_TIME     ’加減速データ指定方法:時間
  spd.Acc(axNo) = 0.5                     ’加速時間:0.5Sec
  spd.Dec(axNo) = 0.5                     ’減速時間:0.5Sec
  spd.SMode(axNo) = MNET_NON_S            ’S字未使用

  MNETM4SetSpeedNo(lineNo, devNo, axNo, spd, MNET_STAMODE_UD)  ‘速度データ設定

  ’移動データ構造体にデータ設定
  dst.Pos(axNo) = 10000                   ’移動量:10000パルス
  dst.StartMode = MNET_STAMODE_UD         ’スタートモード:加減速

  MNETM4PtpIncNo(lineNo, devNo, axNo, dst)  ’PtpInc移動
```

```
’位置決め完了を待つ
axPtn = &H1         ’位置決め完了待ちをする軸パターンデータ設定

Do While (posEnd = False)
MNETM4StopChk(lineNo, devNo, axPtn, stopSts ’動作停止確認)
   If (stopSts<>0) Then
    posEnd = True
  EndIf
Loop

MNETCenterReset(lineNo)                    ’センターデバイスリセット
PPCI_L112Close()                           ’ボードのクローズ
```

7−3. データ定義

7−3−1. ビットデータ構造体

ビットデータの読出しに使用します。(MNETM4GetBitDataStruct 関数使用時)
ビットデータを一括で取得することが可能です。

書式

[VisualC++]

```
//ビットデータ取得用構造体
struct TMNETM4BitData {
  short P0;         // PORT-Bit0
  short P1;         // PORT-Bit1
  short P2;         // PORT-Bit2
  short P3;         // PORT-Bit3

  short Srun;       // SRUN(動作中)
  short Send;       // SEND(動作停止)
  short Serr;       // SERR(エラー割り込み発生)
  short Sint;       // SINT(イベント割込み発生)
  short Scp1;       // コンパレータ1比較条件成立
  short Scp2;       // コンパレータ2比較条件成立
  short Sprf;       // プリレジフル
  short Salm;       // ALM入力
  short Spel;       // PEL(プラスエンドリミット)入力
  short Smel;       // MEL(マイナスエンドリミット)入力
  short Sorg;       // ORG入力

  short Cnd;        // 軸動作状態
  short Scd;        // CSD入力
  short Ssta;       // STA入力
  short Sstp;       // STP入力
  short Semg;       // EMG入力
  short Spcs;       // PCS入力
  short Serc;       // ERC出力
  short Sez;        // EZ入力
  short Sdrp;       // +DR(PA)信号状態
  short Sdrm;       // -DR(PB)信号状態
  short Sltc;       // LTC入力
  short Sdin;       // SD入力
  short Sinp;       // INP入力
  short Sdir;       // 動作方向
};
```


Motionnet®
Remote I/O & Remote Motion

[VisualBasic]

```
Type TMNETM4BitData
  P0 As Integer          '/ PORT-Bit0
  P1 As Integer          '/ PORT-Bit1
  P2 As Integer          '/ PORT-Bit2
  P3 As Integer          '/ PORT-Bit3

  Srun As Integer        '/ SRUN(動作中)
  Send As Integer        '/ SEND(動作停止)
  Serr As Integer        '/ SERR(エラー割り込み発生)
  Sint As Integer        '/ SINT(イベント割込み発生)
  Scp1 As Integer        '/ コンパレータ1比較条件成立
  Scp2 As Integer        '/ コンパレータ2比較条件成立
  Sprf As Integer        '/ プリレジフル
  Salm As Integer        '/ ALM入力
  Spel As Integer        '/ PEL(プラスエンドリミット)入力
  Smel As Integer        '/ MEL(マイナスエンドリミット)入力
  Sorg As Integer        '/ ORG入力

  Cnd As Integer         '/ 軸動作状態
  Scd As Integer         '/ CSD入力
  Ssta As Integer        '/ STA入力
  Sstp As Integer        '/ STP入力
  Semg As Integer        '/ EMG入力
  Spcs As Integer        '/ PCS入力
  Serc As Integer        '/ ERC出力
  Sez As Integer         '/ EZ入力
  Sdrp As Integer        '/ +DR(PA)信号状態
  Sdrm As Integer        '/ -DR(PB)信号状態
  Sltc As Integer        '/ LTC入力
  Sdin As Integer        '/ SD入力
  Sinp As Integer        '/ INP入力
  Sdir As Integer        '/ 動作方向
End Type
```

[VB.NET]

```
Structure TMNETM4BitData
  P0 As Short          '/ PORT-Bit0
  P1 As Short          '/ PORT-Bit1
  P2 As Short          '/ PORT-Bit2
  P3 As Short          '/ PORT-Bit3

  Srun As Short         '/ SRUN(動作中)
  Send As Short         '/ SEND(動作停止)
  Serr As Short         '/ SERR(エラー割り込み発生)
  Sint As Short         '/ SINT(イベント割込み発生)
  Scp1 As Short         '/ コンパレータ1比較条件成立
  Scp2 As Short         '/ コンパレータ2比較条件成立
  Sprf As Short         '/ プリレジフル
  Salm As Short         '/ ALM入力
  Spel As Short         '/ PEL(プラスエンドリミット)入力
  Smel As Short         '/ MEL(マイナスエンドリミット)入力
  Sorg As Short         '/ ORG入力

  Cnd As Short          '/ 軸動作状態
  Scd As Short          '/ CSD入力
  Ssta As Short         '/ STA入力
  Sstp As Short         '/ STP入力
  Semg As Short         '/ EMG入力
  Spcs As Short         '/ PCS入力
  Serc As Short         '/ ERC出力
  Sez As Short          '/ EZ入力
  Sdrp As Short         '/ +DR(PA)信号状態
  Sdrm As Short         '/ -DR(PB)信号状態
  Sltc As Short         '/ LTC入力
  Sdin As Short         '/ SD入力
  Sinp As Short         '/ INP入力
  Sdir As Short         '/ 動作方向
End Structure
```

[メンバー]

メンバ	内容	取得値
P0	汎用I/Oポート P0状態	0:OFF, 1:ON
P1	汎用I/Oポート P1状態	0:OFF, 1:ON
P2	汎用I/Oポート P2状態	0:OFF, 1:ON
P3	汎用I/Oポート P3状態	0:OFF, 1:ON
Srun	Srunビット状態(動作中)	0:OFF, 1:ON
Send	Sendビット状態(動作停止)	0:OFF, 1:ON
Serr	Serrビット状態(エラー割込有)	0:OFF, 1:ON
Sint	Sintビット状態(イベント割込有)	0:OFF, 1:ON
Scp1	CMP1比較条件成立状態	0:非成立, 1:成立
Scp2	CMP2比較条件成立状態	0:非成立, 1:成立
Sprf	次動作用プリレジスタ満杯	0:空き,1:満杯
Salm	ALM信号状態	0:OFF, 1:ON
Spel	+EL信号状態	0:OFF, 1:ON
Smel	-EL信号状態	0:OFF, 1:ON
Sorg	ORG信号状態	0:OFF, 1:ON
Cnd	軸動作状態	下表参照
Scd	CSD信号状態	0:OFF, 1:ON
Ssta	CSTA信号状態	0:OFF, 1:ON
Sstp	CSTP信号状態	0:OFF, 1:ON
Semg	EMG信号状態	0:OFF, 1:ON
Spcs	PCS信号状態	0:OFF, 1:ON
Serc	ERC信号状態	0:OFF, 1:ON
Sez	EZ信号状態	0:OFF, 1:ON
Sdrp	+DR(PA)信号状態	0:OFF, 1:ON
Sdrm	-DR(PB)信号状態	0:OFF, 1:ON
sltc	LTC信号状態	0:OFF, 1:ON
Sdin	SD信号状態	0:OFF, 1:ON
sinp	INP信号状態	0:OFF, 1:ON
sdir	動作方向	0:+方向, 1:-方向

信号状態は論理パラメータが加味された結果となります。
またこの値はLSIとしての値となります。ユニットとしての信号論理は各I/F回路を御確認ください。

cnd(軸動作状態)一覧

取得値	状態	取得値	状態
0	停止中	8	PA/PB入力待ち
1	DR入力待ち	9	(スタート/ストップ制御中)
2	CSTA入力待ち	10	FL定速動作中
3	内部同期信号待ち	11	加速中
4	他軸の停止待ち	12	FH定速動作中
5	ERCタイマ完了待ち	13	減速中
6	方向変化タイマ完了待ち	14	INP入力待ち
7	(スタート/ストップ制御中)	15	(スタート/ストップ制御中)

7−3−2. ステータス／ビットマップ

7−3−2−1. エラー割り込み要因

エラー割り込み要因のマップです。
エラー割り込みが発生したときに対応するビットが 1 になります。
データの取得は MNETM4GetREST 関数で行います。
このデータはパルスコントロールLSI:PCL6143のREST レジスタ(エラー割り込みステータス)から取得しています。このレジスタは読出しによりリセットされます。

Bit	ﾋﾞｯﾄ名	内容
0	ESPL	＋EL入力ONによる停止時
1	ESML	－EL入力ONによる停止時
2	ESAL	ALM入力ONによる停止時。または停止中のALM入力
3	ESSP	CSTP入力ONによる停止時
4	ESEM	EMG入力ONによる停止時。または停止中のEMG入力
5	ESSD	SD入力ONによる減速停止時
6	ESPO	PA/PB入力用バッファカウンタのオーバーフロー発生による停止時
7	ESEE	EA/EB入力エラー発生時
8	ESPE	PA/PB入力エラー発生時
31～9	未定義	未定義（常に0になります。）

7−3−2−2. イベント割り込み要因

イベント割り込み要因のマップです。
MNETM4SetPar 関数で設定したイベントが発生したときに対応するビットが 1 になります。
データの取得は MNETM4GetRIST 関数で行います。
このデータはパルスコントロールLSI:PCL6143のRIST レジスタ(イベント割り込みステータス)から取得しています。このレジスタは読出しによりリセットされます。

Bit	ﾋﾞｯﾄ名	内容
0	ISEN	自動停止時
1	ISNM	プリレジスタ書込み可能時
2	ISUS	加速開始時
3	ISUE	加速終了時
4	ISDS	減速開始時
5	ISDE	減速終了時
6	ISC1	コンパレータ 1 条件成立時
7	ISC2	コンパレータ 2 条件成立時
8	ISLT	LTC 入力によるカウント値のラッチ時
9	ISOL	ORG 入力 ON 時
10	ISSD	SD 入力 ON 時
11	ISPD	+DR(PA)入力変化時
12	ISMD	-DR(PB)入力変化時
13	ISSA	CSTA 入力 ON 時
31～14	未定義	未定義（常に 0 になります。）

7−3−3. 識別子

7−3−3−1. スタートモード

スタートモードは下記3パターンがあります。

MNETM4SetSpeed関数での速度データ設定時に速度データ構造体のメンバー「StartMode」に、および移動命令関数の引数に設定するデータとその識別子は次のようになります。

識別子	値	内容
MNET_STAMODE_FL	0050h	FL定速
MNET_STAMODE_FH	0051h	FH 定速
MNET_STAMODE_UD	0053h	加減速

7−3−3−2. 動作方向

連続動作や原点復帰動作など、動作方向を指定する移動命令の場合に、速度データ構造体のメンバー「Dir」に設定します。

識別子	値	内容
MNET_DIR_PLUS	0	プラス方向
MNET_DIR_MINUS	1	マイナス方向

7−3−3−3. 加減速指定方法

MNETM4SetSpeed関数などでの加減速データ設定時､加減速度を加速(減速)時間で指定するか､加速(減速)レート(1秒あたりの増分(減分)パルス)で指定するかを、速度データ構造体のメンバー「AdcMtd」に設定します。

識別子	値	内容
MNET_ADC_BY_RATE	0	レート指定(1秒あたりの増分(減分)パルス速度)
MNET_ADC_BY_TIME	1	時間指定　(秒)

7−3−3−4. S 字使用/未使用

MNETM4SetSpeed関数などでの速度データ設定時、S字動作を使用するかどうかを、速度データ構造体のメンバー「SMode」に設定します。

識別子	値	内容
MNET_NON_S	0	S字動作なし
MNET_USE_S	1	S字動作あり

7−3−3−5. カウンタ指定

モーションデバイスのカウンタデータ取得/設定時に、対象カウンタを指定します。

識別子	値	内容
MNET_SEL_COUNTER1	0	カウンタ1選択
MNET_SEL_COUNTER2	1	カウンタ2選択
MNET_SEL_COUNTER3	2	カウンタ3選択

7−3−3−6. ラッチ指定

モーションデバイスのラッチデータ取得時に、対象のラッチを指定します。

識別子	値	内容
MNET_SEL_LTC1	0	ラッチ1選択
MNET_SEL_LTC2	1	ラッチ2選択
MNET_SEL_LTC3	2	ラッチ3選択

7−3−3−7. 停止モード

MNETM4Stop関数使用時に停止方法を指定します。

識別子	値	内容
MNET_STOP	0	即停止
MNET_SDSTOP	1	減速停止
MNET_EMGSTOP	2	非常停止

7−3−3−8. 速度倍率レンジ

識別子	値	内容
MNETM4_SPEED_03_RANGE	0	速度倍率:0.3倍
MNETM4_SPEED_05_RANGE	1	速度倍率:0.5倍
MNETM4_SPEED_1_RANGE	2	速度倍率:1倍
MNETM4_SPEED_2_RANGE	3	速度倍率:2倍
MNETM4_SPEED_5_RANGE	4	速度倍率:5倍
MNETM4_SPEED_10_RANGE	5	速度倍率:10倍
MNETM4_SPEED_20_RANGE	6	速度倍率:20倍
MNETM4_SPEED_50_RANGE	7	速度倍率:50倍
MNETM4_SPEED_100_RANGE	8	速度倍率:100倍
MNETM4_SPEED_200_RANGE	9	速度倍率:200倍
MNETM4_SPEED_400_RANGE	10	速度倍率:400倍
MNETM4_SPEED_600_RANGE	11	速度倍率:600倍

7−3−3−9. ビット選択

MNETM4GetBitData, MNETM4SetBitData 関数使用時に対象ビットを指定します。

識別子	値	内容	取得(設定値)
MNETM4_SEL_BIT_P0	0	汎用I/Oポート P0信号状態	0:OFF, 1:ON **(Set可能)**
MNETM4_SEL_BIT_P1	1	汎用I/Oポート P1信号状態	0:OFF, 1:ON **(Set可能)**
MNETM4_SEL_BIT_P2	2	汎用I/Oポート P2信号状態	0:OFF, 1:ON **(Set可能)**
MNETM4_SEL_BIT_P3	3	汎用I/Oポート P3信号状態	0:OFF, 1:ON **(Set可能)**
MNETM4_SEL_BIT_SRUN	10	SRUNビット状態(動作中)	0:OFF, 1:ON
MNETM4_SEL_BIT_SEND	11	SENDビット状態(動作停止)	0:OFF, 1:ON
MNETM4_SEL_BIT_SERR	12	SERRビット状態(エラー割込有)	0:OFF, 1:ON
MNETM4_SEL_BIT_SINT	13	SINTビット状態(イベント割込有)	**0:OFF, 1:ON**
MNETM4_SEL_BIT_SCP1	14	CMP1比較条件成立状態	0:非成立, 1:成立
MNETM4_SEL_BIT_SCP2	15	CMP2比較条件成立状態	0:非成立, 1:成立
MNETM4_SEL_BIT_SPRF	16	プリレジスタ満杯	0:空き, 1:満杯
MNETM4_SEL_BIT_SALM	17	ALM信号状態	0:OFF, 1:ON
MNETM4_SEL_BIT_SPEL	18	+EL信号状態	0:OFF, 1:ON
MNETM4_SEL_BIT_SMEL	19	-EL信号状態	0:OFF, 1:ON
MNETM4_SEL_BIT_SORG	20	ORG信号状態	0:OFF, 1:ON
MNETM4_SEL_BIT_CND	21	CND(軸動作状態)	7-3-1.ビットデータ構造体参照
MNETM4_SEL_BIT_SCD	22	CSD信号状態	0:OFF, 1:ON
MNETM4_SEL_BIT_SSTA	23	CSTA信号状態	0:OFF, 1:ON
MNETM4_SEL_BIT_SSTP	24	CSTP信号状態	0:OFF, 1:ON
MNETM4_SEL_BIT_SEMG	25	EMG信号状態	0:OFF, 1:ON
MNETM4_SEL_BIT_SPCS	26	PCS信号状態	0:OFF, 1:ON
MNETM4_SEL_BIT_SERC	27	ERC信号状態	0:OFF, 1:ON **(Set可能)**
MNETM4_SEL_BIT_SEZ	28	EZ信号状態	0:OFF, 1:ON
MNETM4_SEL_BIT_SDRP	29	+DR(PA)信号状態	0:OFF, 1:ON
MNETM4_SEL_BIT_SDRM	30	-DR(PB)信号状態	0:OFF, 1:ON
MNETM4_SEL_BIT_SLTC	31	LTC信号状態	0:OFF, 1:ON
MNETM4_SEL_BIT_SDIN	32	SD信号状態	0:OFF, 1:ON
MNETM4_SEL_BIT_SINP	33	INP信号状態	0:OFF, 1:ON
MNETM4_SEL_BIT_SDIR	34	動作方向	0:+方向, 1:-方向

信号状態は論理パラメータが加味された結果となります。
またこの値はLSIとしての値となります。ユニットとしての信号論理は各I/F回路を御確認ください。

7−3−3−10. パラメータ項目番号定義

MNETM4GetPar関数, MNETM4SetPar関数使用時に対象のパラメータ項目を指定します。

識別子	値	内容	設定範囲	設定値／取得値 （網掛け：デフォルト(リセット時)）
MNETM4_PAR_BASE_DATA	400h	基本固定パラメータ(注1)	0	0を設定ください
MNETM4_PAR_OUTPLS_TYPE	401h	出力パルス仕様	0～7	当項下部記載の出力パルス仕様参照 0
MNETM4_PAR_DIR_TIM_ENA	402h	方向変化タイマ無効	0/1	0:有効, 1:無効
MNETM4_PAR_SDLTC_ENA	411h	SDラッチ有効	0/1	0:無効, 1:有効
MNETM4_PAR_SD_LOGIC	412h	SD信号論理	0/1	0:負論理, 1:正論理
MNETM4_PAR_ORG_LOGIC	414h	ORG信号論理	0/1	0:負論理, 1:正論理
MNETM4_PAR_ALM_LOGIC	415h	ALM信号論理	0/1	0:負論理, 1:正論理
MNETM4_PAR_ALM_PROC	416h	ALM入力時処理	0/1	0:即停止, 1:減速停止
MNETM4_PAR_ERC_LOGIC	417h	ERC信号論理	0/1	0:負論理, 1:正論理
MNETM4_PAR_ERCPLS_LEN	418h	ERCパルス幅	0～7	0:12μs, 1:102μs, 2:409μs 3:1.6ms 4:13ms, 5:52ms, 6:104ms, 7:ﾚﾍﾞﾙ出力
MNETM4_PAR_ERR_ERC_PROC	419h	エラー停止時ERC出力	0/1	0:しない, 1:出力する
MNETM4_PAR_ORG_ERC_PROC	41Ah	原点復帰時ERC出力	0/1	0:しない, 1: 出力する
MNETM4_PAR_INP_LOGIC	41Ch	INP信号論理	0/1	0:負論理, 1:正論理
MNETM4_PAR_PCS_LOGIC	41Eh	PCS信号論理	0/1	0:負論理, 1:正論理
MNETM4_PAR_LTC_LOGIC	41Fh	LTC信号論理	0/1	0:立下り, 1:立上り
MNETM4_PAR_LTC1_INP_DIS	420h	LTC1ハード入力無効	0/1	0:有効,1:無効
MNETM4_PAR_LTC2_INP_DIS	421h	LTC2ハード入力無効	0/1	0:有効,1:無効
MNETM4_PAR_EL_PROC	423h	EL入力時処理	0/1	0:即停止, 1:減速停止 (注2,3)
MNETM4_PAR_INP_FILT_ENA	424h	入力フィルタ	0/1	0:無効, 1:有効 (注4)
MNETM4_PAR_INP_FILT_MODE	425h	入力フィルタ仕様	0～3	0:3.2μs以下の入力は無視 1:25μs以下の入力は無視 2:200μs以下の入力は無視 3:1.6ms以下の入力は無視
MNETM4_PAR_ENC_TYPE	430h	エンコーダ仕様	0～3	0:90度位相差1逓倍 1:90度位相差2逓倍 2:90度位相差4逓倍 3:EAの立上りでカウントアップ, EBの立上りでカウントダウン
MNETM4_PAR_ENC_REV	431h	エンコーダ入力反転	0/1	0:なし, 1:反転
MNETM4_PAR_ENC_FILT_ENA	432h	エンコーダフィルタ	0/1	0:無効, 1:有効 (注4)
MNETM4_PAR_ENC_DIS	433h	エンコーダ入力無効	0/1	0:有効, 1:無効
MNETM4_PAR_PULSER_TYPE	440h	パルサー仕様	0～3	0:90度位相差1逓倍 1:90度位相差2逓倍 2:90度位相差4逓倍 3:PAの立上りでカウントアップ, PBの立上りでカウントダウン
MNETM4_PAR_PULSER_REV	441h	パルサー入力反転	0/1	0:無効, 1:有効
MNETM4_PAR_PULSER_FILT_ENA	444h	パルサーフィルタ	0/1	0:無効, 1:有効 (注4)
MNETM4_PAR_PULSER_DIS	445h	パルサー入力無効	0/1	0:有効, 1:無効

注１．本 DLL の仕様により決まっている固定パラメータ設定です。 本 DLL 使用時には必ず設定ください。

注 2. EL 入力 ON 時の処理を減速停止に設定した場合、EL 入力 ON で減速を開始しますので、EL 位置を通過して停止します。機械系への衝突等に十分御注意下さい。

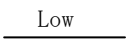
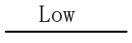
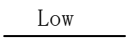
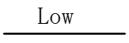
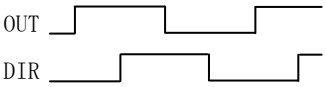
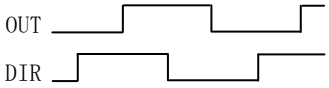
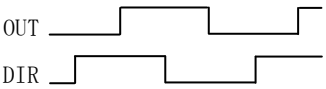
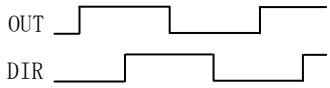
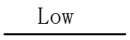
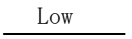
注 3. 停止方法を減速停止に設定した場合には、減速して停止するまで入力 ON 状態を保持して下さい。
正常停止と異常停止との判別は停止タイミングの後で決定されます。そのため、加減速位置決め動作で減速中に異常停止信号が入力されても正常停止と認識する場合があります。
その場合でも異常停止信号を入力し続けていれば、次動作は継続せずに異常停止になります。

注 4. リセット時は無効ですが、基本固定パラメータ設定により有効に設定されます。

識別子	値	内容	設定範囲	設定値／取得値 （網掛け：デフォルト(リセット時)）
MNETM4_PAR_ORG_MODE	450h	原点復帰方法	0/1	原点復帰方法説明の項参照 0
MNETM4_PAR_EZ_CNT	451h	Z相回数	0～15	0(1回)～15(16回)となります。0
MNETM4_PAR_EZ_LOGIC	452h	Z相論理	0/1	0:立下り, 1:立上り
MNETM4_PAR_CMP1_MODE	461h	コンパレータ1比較 方法	0～3	0:コンパレータ機能OFF 1:コンパレータ1 = カウンタ1 2:コンパレータ1 > カウンタ1 3:コンパレータ1 < カウンタ1
MNETM4_PAR_CMP1_DATA	463h	コンパレータ1データ	右記	-134,217,728～134,217,727 (8000000h～7ffffffh) 0
MNETM4_PAR_CMP2_MODE	465h	コンパレータ2比較 方法	0～3	0:コンパレータ機能OFF 1:コンパレータ2 = カウンタ2 2:コンパレータ2 > カウンタ2 3:コンパレータ2 < カウンタ2
MNETM4_PAR_CMP2_DATA	467h	コンパレータ2データ	右記	-134,217,728～134,217,727 (8000000h～7ffffffh) 0
MNETM4_PAR_SET_EVTINT	470h	イベント割込設定	右記	0h～1fffh 0
MNETM4_PAR_SET_CSTA_TYPE	481h	CSTA入力仕様	0/1	0:レベルトリガ, 1:エッジトリガ
MNETM4_PAR_SET_CSTP_PROC	484h	CSTP入力時処理	0/1	0:即停止, 1:減速停止
MNETM4_PAR_SET_SYNC_OUT_ MODE	485h	内部同期信号出力タイミング	0～15	1:コンパレータ1条件成立時 2:コンパレータ2条件成立時 8:加速開始時 9:加速終了時 10:減速開始時 11:減速終了時 その他:内部同期信号出力OFF
MNETM4_PAR_SET_SYNC_INP	486h	内部同期信号入力選択	0～3	0:軸0が出力した内部同期信号 1:軸1が出力した内部同期信号 2:軸2が出力した内部同期信号 3:軸3が出力した内部同期信号
MNETM4_PAR_PIO2_TYPE	490h	汎用I/O(P2)仕様	0～3	0:汎用入力, 1:汎用出力 2:定速動作中信号を負論理出力 3:定速動作中信号を正論理出力
MNETM4_PAR_PIO3_TYPE	491h	汎用I/O(P3)仕様	0～3	0:汎用入力, 1:汎用出力 2: CP1状態を負論理で出力 3: CP1状態を正論理で出力
MNETM4_PAR_FH_CORRECT_DIS	4F1h	FH補正有効/無効	0/1	0:有効, 1:無効
MNETM4_PAR_SET_CSTP_ENA	4F2h	CSTP有効	0/1	0:無効, 1:有効
MNETM4_PAR_SET_AUTO_CSTA_ ENA	4F3h	CSTP自動出力有効	0/1	0:無効, 1:有効 (異常停止時にCSTP自動出力)
MNETM4_PAR_SET_CSD_ENA	4F4h	CSD有効	0/1	0:無効, 1:有効
MNETM4_PAR_SET_AUTO_CSD_ENA	4F5h	CSD自動出力有効	0/1	0:無効, 1:有効 (減速時,FL定速動作時にCSD自動出力)
MNETM4_PAR_PCS_ENA	4F6h	PCS信号有効	0/1	0:無効, 1:有効
MNETM4_PAR_SD_ENA	4F7h	SD有効	0/1	0:無効, 1:有効
MNETM4_PAR_INP_ENA	4F8h	INP信号有効	0/1	0:無効, 1:有効
MNETM4_PAR_MSY_MODE	4F9h	同期スタート	0～3	0:無効 1:同時スタート(CSTA)信号待ち 2:内部同期スタート信号待ち 3:指定軸の停止待ち
MNETM4_PAR_MSY_SAX	4FAh	他軸停止スタート設定	0～15	停止確認する軸パターンを4ビットで 設定します。デフォルト:0h ビット0:軸0の停止を確認します ビット1:軸1の停止を確認します ビット2:軸2の停止を確認します ビット3:軸3の停止を確認します

出力パルスタイプ

出力パルス仕様を設定します。

設定/取得値	（＋）方向動作時		（－）方向動作時	
	PULS 信号	DIR 信号	PULS 信号	DIR 信号
0				
1				
2				
3				
4				
5				
6				
7				

7－3－3－11. DLL 内部パラメータ項目番号定義

MNETM4SetDLLPar関数使用時に対象のパラメータ項目を指定します。

識別子	値	内容	設定範囲	設定値／取得値（網掛け：デフォルト）
MNETM4_DLLPAR_PRECHK_DIS	4E0h	軸動作前確認省略	0/1	0:省略しない,1:省略
MNETM4_DLLPAR_ADCCR_DIS	4E1h	加減速設定自動補正	0/1	0:補正する,1:補正しない

Motionnet®
Remote I/O & Remote Motion

7−3−4. 戻り値

識別子	値	内容	備考
MNET_ERR_M4_SET_ADDR	400Fh	アドレス指定不正	
MNET_ERR_M4_SET_SMODE	4020h	S字モード指定エラー	(注1)
MNET_ERR_M4_SET_DIR	4030h	方向指定エラー	(注1)
MNET_ERR_M4_SET_ADCMTD	4040h	加減速方法指定エラー	(注1)
MNET_ERR_M4_SET_SPD_DATA	4050h	速度指定エラー	(注1)
MNET_ERR_M4_SET_ADC_DATA	4060h	加減速度指定エラー	(注1)
MNET_ERR_M4_SET_SSECT	4070h	S字範囲指定エラー	(注1)
MNET_ERR_M4_CALC_ACC_DATA	4080h	加速レート計算結果エラー	(注1)
MNET_ERR_M4_CALC_DEC_DATA	4090h	減速レート計算結果エラー	(注1)
MNET_ERR_M4_CALC_SPD_DATA	40A0h	速度計算結果エラー	(注1)
MNET_ERR_M4_STATUS_NOT_ READY	4100h	受付状態にない(動作中など)	(注1)
MNET_ERR_M4_STATUS_ALM	4110h	受付状態にない(ALM)	(注1)
MNET_ERR_M4_STATUS_EL	4120h	受付状態にない(EL)	(注1)
MNET_ERR_M4_STATUS_CSTP	4130h	受付状態にない(CSTP)	(注1)
MNET_ERR_M4_STATUS_ALREADY	4140h	すでに目的の状態にある	(注1)
MNET_ERR_M4_FAILED_OVRSPD	4160h	速度オーバーライド失敗	(注1)
MNET_ERR_M4_STATUS_SERR	4170h	エラー割込あり	(注1)
MNET_ERR_M4_STATUS_PREREG_ FULL	4180h	プリレジスタ満杯	(注1)
MNET_ERR_M4_SPDDATA_INVALID	4190h	速度データ無効	(注1)
MNET_ERR_DEV_NOT_M4	450Fh	デバイスが4軸モーションデバイスでない	
MNET_ERR_M4_SET_STAMODE	451Fh	スタートモード指定エラー	
MNET_ERR_M4_SET_ INTERPOLATION	452Fh	補間指令データ不正	
MNET_ERR_M4_SET_AXNO	453Fh	軸番号指定不正	
MNET_ERR_M4_SET_AXPTN	454Fh	軸パターン指定不正	
MNET_ERR_M4_SET_PREREG	455Fh	プリレジモード指定エラー	
MNET_ERR_M4_PREREG_CONTINUE	456Fh	プリレジ継続不可	
MNET_ERR_M4_STATUS_EMG	457Fh	受付状態にない(EMG)	
MNET_ERR_M4_PCS_MODE	458Fh	PCS有効では受付不可	
MNET_ERR_M4_SET_SYNC_LINE	459Fh	補間軸間で同期スタートモードが異なる	
MNET_ERR_M4_FAILED_FILEOPEN	45AFh	ファイルオープン失敗	
MNET_ERR_M4_INVALID_FILEDATA	45BFh	ファイルデータ不正	

(注 1)

値の 16 進値最下位桁が F 以外の場合、値に軸番号(0～3)が最下位桁に付加(OR)されます。

[例 1]

値　4032h は 4030h(エラー内容を示すコード) + 2(軸番号)を表し、軸 2 の方向指定エラーを示します。

[例 2]

値　4180h は 4180h(エラー内容を示すコード) + 0(軸番号)を表し、軸 0 のプリレジ満杯を示します。

Motionnet®
Remote I/O & Remote Motion

8. ローカル 4 軸(6045)モーションボード機能

8−1. 関数詳細

関数詳細説明にあたって

(1) 軸パターンについて

軸パターンとは、4 軸を下記のように 4 ビットのデータで表し、複数軸を同時に対象として選択するデータです。

[例]

軸 0 と軸 2 の場合 ： 5h

軸3	軸2	軸1	軸0
0	1	0	1

軸 0 の場合 ： 1h

軸3	軸2	軸1	軸0
0	0	0	1

軸 0 と軸 2 と軸 3 の場合 ： Dh

軸3	軸2	軸1	軸0
1	1	0	1

Motionnet®
Remote I/O & Remote Motion

8－1－1. ローカル４軸(6045)モーションボード関数

8－1－1－1. MNET6045SetSpeed

機能

（非補間動作）速度データ設定

- 本関数により設定された速度データは以降の非補間動作の動作速度として適用されます。
- 補間動作実行により、該当軸の速度データは無効となりますので、非補間動作実行前には改めて速度データの設定が必要となります。

書式

[Visual C++]

```
short __stdcall MNET6045SetSpeed(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Spd* spd,
                    short startMode);
```

[Visual Basic]

```
Declare Function MNET6045SetSpeed Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef spd As TMNET6045Spd,
                    ByVal startMode As Integer)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045SetSpeed Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef spd As TMNET6045Spd,
                    ByVal startMode As Integer) As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axPtn	***軸パターン***（8-2-1.軸パターン参照）
spd	***速度データ構造体***（8-2-4.速度データ構造体参照）
startMode	***スタートモード***（8-2-3.スタートモード参照）

戻り値

正常に終了した場合は、MNET_SUCCESS が返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

[例1]

```
TMNET6045Spd spd = {0};              // 速度データ構造体定義,初期化

spd.SpdL[0] = 2000;                  // FL速度設定
spd.SpdH[0] = 10000;                 // FH速度設定
spd.Acc[0] = 1.5;                    // 加速データ
spd.Dec[0] = 5;                      // 減速データ
spd.AdcSpc[0] = MNET_ADC_BY_TIME;    // 加減速指定方法: 時間(秒)

MNET6045SetSpeed(0, 0, 0x1, &spd, MNET_STAMODE_UD);
```

ライン0、デバイス0、軸パターン1h(軸0)の速度データを設定します。

[例2]

```
TMNET6045Spd spd = {0};              // 速度データ構造体定義,初期化

spd.SpdL[1] = 2000;                  // FL速度設定
spd.SpdH[1] = 10000;                 // FH速度設定
spd.Acc[1] = 1000;                   // 加速データ
spd.Dec[1] = 2000;                   // 減速データ
spd.AdcSpc[1] = MNET_ADC_BY_RATE;    // 加減速指定方法: レート(秒あたりの増減分パルス)
spd.SMode[1] = MNET_USE_S;           // S字使用

MNET6045SetSpeed(0, 0, 0x2, &spd, MNET_STAMODE_UD);
```

ライン0、デバイス0、軸パターン2h(軸1)の速度データを設定します。

[例3]

```
TMNET6045Spd spd = {0};              // 速度データ構造体定義,初期化

spd.SpdL[0] = 2000;                  // FL速度設定
spd.SpdH[0] = 10000;                 // FH速度設定
spd.Acc[0] = 1.5;                    // 加速データ
spd.Dec[0] = 5;                      // 減速データ
spd.AdcSpc[0] = MNET_ADC_BY_TIME;    // 加減速指定方法: 時間(秒)
spd.SpdL[2] = 10000;                 // FL速度設定
spd.SpdH[2] = 10000;                 // FH速度設定
spd.Acc[2] = 1000;                   // 加速データ
spd.Dec[2] = 2000;                   // 減速データ
spd.AdcSpc[2] = MNET_ADC_BY_RATE;    // 加減速指定方法: レート(秒あたりの増減分パルス)
spd.SMode[2] = MNET_USE_S;           // S字使用

MNET6045SetSpeed(0, 0, 0x5, &spd, MNET_STAMODE_UD);
```

ライン0、デバイス0、軸パターン5h(軸0+軸2)の速度データを設定します。

[Visual Basic]

[例1]

```
Dim spd As TMNET6045Spd            '/ 速度データ構造体定義,初期化

spd.SpdL(0) = 2000                 '/ FL速度設定
spd.SpdH(0) = 10000                '/ FH速度設定
spd.Acc(0) = 1.5                   '/ 加速データ
spd.Dec(0) = 5                     '/ 減速データ
spd.AdcSpc(0) = MNET_ADC_BY_TIME   '/ 加減速指定方法: 時間(秒)

MNET6045SetSpeed 0, 0, &H1, spd, MNET_STAMODE_UD
```

ライン0、デバイス0、軸パターン1h(軸0)の速度データを設定します。

[例2]

```
Dim spd As TMNET6045Spd            '/ 速度データ構造体定義,初期化

spd.SpdL(1) = 2000                 '/ FL速度設定
spd.SpdH(1) = 10000                '/ FH速度設定
spd.Acc(1) = 1000                  '/ 加速データ
spd.Dec(1) = 2000                  '/ 減速データ
spd.AdcSpc(1) = MNET_ADC_BY_RATE   '/ 加減速指定方法: レート(秒あたりの増減分パルス)
spd.SMode(1) = MNET_USE_S          '/ S字使用

MNET6045SetSpeed 0, 0, &H2, spd, MNET_STAMODE_UD
```

ライン0、デバイス0、軸パターン2h(軸1)の速度データを設定します。

[例3]

```
Dim spd As TMNET6045Spd            '/ 速度データ構造体定義,初期化

spd.SpdL(0) = 2000                 '/ FL速度設定
spd.SpdH(0) = 10000                '/ FH速度設定
spd.Acc(0) = 1.5                   '/ 加速データ
spd.Dec(0) = 5                     '/ 減速データ
spd.AdcSpc(0) = MNET_ADC_BY_TIME   '/ 加減速指定方法: 時間(秒)
spd.SpdL(2) = 10000                '/ FL速度設定
spd.SpdH(2) = 10000                '/ FH速度設定
spd.Acc(2) = 1000                  '/ 加速データ
spd.Dec(2) = 2000                  '/ 減速データ
spd.AdcSpc(2) = MNET_ADC_BY_RATE   '/ 加減速指定方法: レート(秒あたりの増減分パルス)
spd.SMode(2) = MNET_USE_S          '/ S字使用

MNET6045SetSpeed 0, 0, &H5, spd, MNET_STAMODE_UD
```

[VB.NET]

[例1]

```
Dim spd As New TMNET6045Spd          '/ 速度データ構造体定義,初期化
spd.Initialize()                     '/ 構造体初期化

spd.SpdL(0) = 2000                   '/ FL速度設定
spd.SpdH(0) = 10000                  '/ FH速度設定
spd.Acc(0) = 1.5                     '/ 加速データ
spd.Dec(0) = 5                       '/ 減速データ
spd.AdcSpc(0) = MNET_ADC_BY_TIME     '/ 加減速指定方法: 時間(秒)

MNET6045SetSpeed(0, 0, &H1, spd, MNET_STAMODE_UD)
```

ライン0、デバイス0、軸パターン1h(軸0)の速度データを設定します。

[例2]

```
Dim spd As New TMNET6045Spd          '/ 速度データ構造体定義,初期化
spd.Initialize()                     '/ 構造体初期化

spd.SpdL(1) = 2000                   '/ FL速度設定
spd.SpdH(1) = 10000                  '/ FH速度設定
spd.Acc(1) = 1000                    '/ 加速データ
spd.Dec(1) = 2000                    '/ 減速データ
spd.AdcSpc(1) = MNET_ADC_BY_RATE     '/ 加減速指定方法: レート(秒あたりの増減分パルス)
spd.SMode(1) = MNET_USE_S            '/ S字使用

MNET6045SetSpeed(0, 0, &H2, spd, MNET_STAMODE_UD)
```

ライン0、デバイス0、軸パターン2h(軸1)の速度データを設定します。

[例3]

```
Dim spd As New TMNET6045Spd          '/ 速度データ構造体定義,初期化
spd.Initialize()                     '/ 構造体初期化

spd.SpdL(0) = 2000                   '/ FL速度設定
spd.SpdH(0) = 10000                  '/ FH速度設定
spd.Acc(0) = 1.5                     '/ 加速データ
spd.Dec(0) = 5                       '/ 減速データ
spd.AdcSpc(0) = MNET_ADC_BY_TIME     '/ 加減速指定方法: 時間(秒)
spd.SpdL(2) = 10000                  '/ FL速度設定
spd.SpdH(2) = 10000                  '/ FH速度設定
spd.Acc(2) = 1000                    '/ 加速データ
spd.Dec(2) = 2000                    '/ 減速データ
spd.AdcSpc(2) = MNET_ADC_BY_RATE     '/ 加減速指定方法: レート(秒あたりの増減分パルス)
spd.SMode(2) = MNET_USE_S            '/ S字使用

MNET6045SetSpeed(0, 0, &H5, spd, MNET_STAMODE_UD)
```

ライン0、デバイス0、軸パターン5h(軸0+軸2)の速度データを設定します。

8－1－1－2. MNET6045PtpInc

機能

Ptp移動(Inc指定)

- 相対移動量指定のPtp移動をスタートします。
- 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。

書式

[Visual C++]

```
short __stdcall MNET6045PtpInc(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045PtpInc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045PtpInc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン*** (8-2-1.軸パターン参照)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESS が返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};              // 移動データ構造体定義,初期化

dst.Pos[0] = 10000;                  // 軸 0 移動量
dst.Pos[1] = -3000;                  // 軸 1 移動量
dst.StartMode = MNET_STAMODE_FL;     // スタートモード:FL 定速

MNET6045PtpInc(0, 0, 0x3, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst              '/ 移動データ構造体定義,初期化

dst.Pos(0) = 10000                   '/ 軸 0 移動量
dst.Pos(1) = -3000                   '/ 軸 1 移動量
dst.StartMode = MNET_STAMODE_FL      '/ スタートモード:FL 定速

MNET6045PtpInc 0, 0, &H3, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化

dst.Pos(0) = 10000                   '/ 軸 0 移動量
dst.Pos(1) = -3000                   '/ 軸 1 移動量
dst.StartMode = MNET_STAMODE_FL      '/ スタートモード:FL 定速

MNET6045PtpInc(0, 0, &H3, dst)
```

ライン 0、デバイス 0、軸パターン 3h(軸 0+軸 1)の Ptp(Inc)移動を行います。

Motionnet®
Remote I/O & Remote Motion

8－1－1－3. MNET6045Ptp

機能

Ptp移動(Abs指定)

- 絶対移動量指定のPtp移動をスタートします。
- 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。

書式

[Visual C++]

```
short __stdcall MNET6045Ptp(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045Ptp Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045Ptp Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン*** (8-2-1.軸パターン参照)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};              // 移動データ構造体定義,初期化

dst.Pos[0] = 20000;                  // 軸0目標位置
dst.Pos[2] = -6000;                  // 軸2目標位置
dst.StartMode = MNET_STAMODE_UD;     // スタートモード:加減速

MNET6045Ptp(0, 0, 0x5, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst              '/ 移動データ構造体定義,初期化

dst.Pos(0) = 20000                   '/ 軸0目標位置
dst.Pos(2) = -6000                   '/ 軸2目標位置
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速

MNET6045Ptp 0, 0, &H5, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化

dst.Pos(0) = 20000                   '/ 軸0目標位置
dst.Pos(2) = -6000                   '/ 軸2目標位置
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速

MNET6045Ptp(0, 0, &H5, dst)
```

ライン0、デバイス0、軸パターン5h(軸0+軸2)のPtp(Abs)移動を行います。

8－1－1－4. MNET6045LineInc

機能

直線補間移動(Inc指定)

- 速度はベクトル速度で指定します。
- 速度データは軸パターンに関わらず速度データ構造体の0軸目に設定します。
- 合成速度一定設定時には加速データ、減速データを等しくする必要があります。
  (合成速度一定設定はMNET6045SetPar関数で行います)

書式

[Visual C++]

```
short __stdcall MNET6045LineInc(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst,
                    TMNET6045Spd* spd);
```

[Visual Basic]

```
Declare Function MNET6045LineInc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst,
                    ByRef spd As TMNET6045Spd)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045LineInc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst,
                    ByRef spd As TMNET6045Spd)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン*** (8-2-1.軸パターン参照)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)
spd	***速度データ構造体*** (8-2-4.速度データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};              // 移動データ構造体定義,初期化
TMNET6045Spd spd = {0};              // 速度データ構造体定義,初期化

dst.Pos[0] = 10000;                  // 軸0移動量
dst.Pos[2] = -3000;                  // 軸2移動量
dst.StartMode = MNET_STAMODE_UD;     // スタートモード:加減速

spd.SpdL[0] = 2000;                  // FL速度設定
spd.SpdH[0] = 10000;                 // FH速度設定
spd.Acc[0] = 1.5;                    // 加速データ
spd.Dec[0] = 5;                      // 減速データ
spd.AdcSpc[0] = MNET_ADC_BY_TIME;    // 加減速指定方法: 時間(秒)

MNET6045LineInc(0, 0, 0x5, &dst, &spd);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst              '/ 移動データ構造体定義,初期化
Dim spd As TMNET6045Spd              '/ 移動データ構造体定義,初期化

dst.Pos(0) = 10000                   '/ 軸0移動量
dst.Pos(2) = -3000                   '/ 軸2移動量
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速

spd.SpdL(0) = 2000                   '/ FL速度設定
spd.SpdH(0) = 10000                  '/ FH速度設定
spd.Acc(0) = 1.5                     '/ 加速データ
spd.Dec(0) = 5                       '/ 減速データ
spd.AdcSpc(0) = MNET_ADC_BY_TIME     '/ 加減速指定方法: 時間(秒)

MNET6045LineInc 0, 0, &H5, dst, spd
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
Dim spd As New TMNET6045Spd          '/ 速度データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化
spd.Initialize()                     '/ 構造体初期化

dst.Pos(0) = 10000                   '/ 軸0移動量
dst.Pos(2) = -3000                   '/ 軸2移動量
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速
spd.SpdL(0) = 2000                   '/ FL速度設定
spd.SpdH(0) = 10000                  '/ FH速度設定
spd.Acc(0) = 1.5                     '/ 加速データ
spd.Dec(0) = 5                       '/ 減速データ
spd.AdcSpc(0) = MNET_ADC_BY_TIME     '/ 加減速指定方法: 時間(秒)

MNET6045LineInc(0, 0, &H5, dst, spd)
```

ライン0、デバイス0、軸パターン5h(軸0+軸2)のInc指定直線補間移動を行います。

Motionnet®
Remote I/O & Remote Motion

8－1－1－5. MNET6045Line

機能

直線補間移動(Abs指定)

- 速度はベクトル速度で指定します。
- 速度データは軸パターンに関わらず速度データ構造体の0軸目に設定します。
- 合成速度一定設定時には加減データ、減速データを等しくする必要があります。
  (合成速度一定設定はMNET6045SetPar関数で行います)

書式

[Visual C++]

```
short __stdcall MNET6045Line(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst,
                    TMNET6045Spd* spd);
```

[Visual Basic]

```
Declare Function MNET6045Line Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst,
                    ByRef spd As TMNET6045Spd)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045Line Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst,
                    ByRef spd As TMNET6045Spd)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン*** (8-2-1.軸パターン参照)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)
spd	***速度データ構造体*** (8-2-4.速度データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};            // 移動データ構造体定義,初期化
TMNET6045Spd spd = {0};            // 速度データ構造体定義,初期化

dst.Pos[1] = 10000;                // 軸1目標位置
dst.Pos[2] = -3000;                // 軸2目標位置
dst.StartMode = MNET_STAMODE_UD;   // スタートモード:加減速

spd.SpdL[0] = 2000;                // FL速度設定
spd.SpdH[0] = 10000;               // FH速度設定
spd.Acc[0] = 500;                  // 加速データ
spd.Dec[0] = 5000;                 // 減速データ
spd.AdcSpc[0] = MNET_ADC_BY_RATE;  // 加減速指定方法: レート(1秒あたりの増減分パルス)
spd.SMode[0] = MNET_USE_S;         // S字使用

MNET6045LineInc(0, 0, 0x6, &dst, &spd);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst            '/ 移動データ構造体定義,初期化
Dim spd As TMNET6045Spd            '/ 速度データ構造体定義,初期化

dst.Pos(1) = 10000                 '/ 軸1目標位置
dst.Pos(2) = -3000                 '/ 軸2目標位置
dst.StartMode = MNET_STAMODE_UD    '/ スタートモード:加減速

spd.SpdL(0) = 2000                 '/ FL速度設定
spd.SpdH(0) = 10000                '/ FH速度設定
spd.Acc(0) = 500                   '/ 加速データ
spd.Dec(0) = 5000                  '/ 減速データ
spd.AdcSpc(0) = MNET_ADC_BY_RATE   '/ 加減速指定方法: レート(1秒あたりの増減分パルス)
spd.SMode(0) = MNET_USE_S          '/ S字使用

MNET6045LineInc 0, 0, &H6, dst, spd
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
Dim spd As New TMNET6045Spd          '/ 速度データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化
spd.Initialize()                     '/ 構造体初期化

dst.Pos(1) = 10000                   '/ 軸1目標位置
dst.Pos(2) = -3000                   '/ 軸2目標位置
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速
spd.SpdL(0) = 2000                   '/ FL速度設定
spd.SpdH(0) = 10000                  '/ FH速度設定
spd.Acc(0) = 500                     '/ 加速データ
spd.Dec(0) = 5000                    '/ 減速データ
spd.AdcSpc(0) = MNET_ADC_BY_RATE     '/ 加減速指定方法: レート(1秒あたりの増減分パルス)
spd.SMode(0) = MNET_USE_S            '/ S字使用

MNET6045LineInc(0, 0, &H6, dst, spd)
```

ライン0、デバイス0、軸パターン6h(軸1+軸2)のInc指定直線補間移動を行います。

Motionnet®
Remote I/O & Remote Motion

8－1－1－6. MNET6045ArcInc

機能

円弧補間移動(Inc指定)

- 速度データは軸パターンに関わらず速度データ構造体の0軸目に設定します。
- 回転方向データは軸パターンに関わらず移動データ構造体の0軸目に設定します。
- 合成速度一定設定時には加減データ、減速データを等しくする必要があります。
  (合成速度一定設定はMNET6045SetPar関数で行います)

書式

[Visual C++]

```
short __stdcall MNET6045ArcInc(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst,
                    TMNET6045Spd* spd);
```

[Visual Basic]

```
Declare Function MNET6045ArcInc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst,
                    ByRef spd As TMNET6045Spd)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045ArcInc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst,
                    ByRef spd As TMNET6045Spd)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン*** (8-2-1.軸パターン参照)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)
spd	***速度データ構造体*** (8-2-4.速度データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};              // 移動データ構造体
TMNET6045Spd spd = {0};              // 速度データ構造体

dst.Pos[0] = 1000;                   // 軸0目標相対位置
dst.Pos[2] = 300;                    // 軸2目標相対位置
dst.Center[0] = 600;                 // 軸0中心相対位置
dst.Center[2] = 0;                   // 軸2中心相対位置
dst.Direction[0] = MNET_DIR_CW;      // 回転方向:CW
dst.StartMode = MNET_STAMODE_UD;     // スタートモード:加減速

spd.SpdL[0] = 2000;                  // FL速度設定
spd.SpdH[0] = 10000;                 // FH速度設定
spd.Acc[0] = 4000;                   // 加速データ
spd.Dec[0] = 6000;                   // 減速データ
spd.AdcSpc[0] = MNET_ADC_BY_RATE;    // 加減速指定方法: レート(1秒あたりの増減分パルス)

MNET6045ArcInc(0, 0, 0x5, &dst, &spd);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst              '/ 移動データ構造体定義,初期化
Dim spd As TMNET6045Spd              '/ 速度データ構造体定義,初期化

dst.Pos(0) = 1000                    '/ 軸0目標相対位置
dst.Pos(2) = 300                     '/ 軸2目標相対位置
dst.Center(0) = 600                  '/ 軸0中心相対位置
dst.Center(2) = 0                    '/ 軸2中心相対位置
dst.Direction(0) = MNET_DIR_CW       '/ 回転方向:CW
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速

spd.SpdL(0) = 2000                   '/ FL速度設定
spd.SpdH(0) = 10000                  '/ FH速度設定
spd.Acc(0) = 4000                    '/ 加速データ
spd.Dec(0) = 6000                    '/ 減速データ
spd.AdcSpc(0) = MNET_ADC_BY_RATE     '/ 加減速指定方法: レート(1秒あたりの増減分パルス)

MNET6045ArcInc 0, 0, &H5, dst, spd
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
Dim spd As New TMNET6045Spd          '/ 速度データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化
spd.Initialize()                     '/ 構造体初期化

dst.Pos(0) = 1000                    '/ 軸0目標相対位置
dst.Pos(2) = 300                     '/ 軸2目標相対位置
dst.Center(0) = 600                  '/ 軸0中心相対位置
dst.Center(2) = 0                    '/ 軸2中心相対位置
dst.Direction(0) = MNET_DIR_CW       '/ 回転方向:CW
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速

spd.SpdL(0) = 2000                   '/ FL速度設定
spd.SpdH(0) = 10000                  '/ FH速度設定
spd.Acc(0) = 4000                    '/ 加速データ
spd.Dec(0) = 6000                    '/ 減速データ
spd.AdcSpc(0) = MNET_ADC_BY_RATE     '/ 加減速指定方法: レート(1秒あたりの増減分パルス)

MNET6045ArcInc(0, 0, &H5, dst, spd)
```

ライン0、デバイス0、軸パターン5h(軸0+軸2)のInc指定円弧補間移動を行います。

8－1－1－7. MNET6045Arc

機能

円弧補間移動(Abs指定)

- 速度データは軸パターンに関わらず速度データ構造体の0軸目に設定します。
- 回転方向データは軸パターンに関わらず移動データ構造体の0軸目に設定します。
- 合成速度一定設定時には加減データ、減速データを等しくする必要があります。
  (合成速度一定設定はMNET6045SetPar関数で行います)

書式

[Visual C++]

```
short __stdcall MNET6045Arc(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst,
                    TMNET6045Spd* spd);
```

[Visual Basic]

```
Declare Function MNET6045Arc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst,
                    ByRef spd As TMNET6045Spd)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045Arc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst,
                    ByRef spd As TMNET6045Spd)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン*** (8-2-1.軸パターン参照)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)
spd	***速度データ構造体*** (8-2-4.速度データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};            // 移動データ構造体定義,初期化
TMNET6045Spd spd = {0};            // 速度データ構造体定義,初期化

dst.Pos[1] = 1000;                 // 軸1目標位置
dst.Pos[2] = 300;                  // 軸2目標位置
dst.Center[1] = 600;               // 軸1中心位置
dst.Center[2] = 0;                 // 軸2中心位置
dst.Direction[0] = MNET_DIR_CW;    // 回転方向:CW
dst.StartMode = MNET_STAMODE_UD;   // スタートモード:加減速

spd.SpdL[0] = 2000;                // FL速度設定
spd.SpdH[0] = 10000;               // FH速度設定
spd.Acc[0] = 1.5;                  // 加速データ
spd.Dec[0] = 3.0;                  // 減速データ
spd.AdcSpc[0] = MNET_ADC_BY_TIME;  // 加減速指定方法: 時間(秒)

MNET6045Arc(0, 0, 0x6, &dst, &spd);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst            '/ 移動データ構造体定義,初期化
Dim spd As TMNET6045Spd            '/ 速度データ構造体定義,初期化

dst.Pos(1) = 1000                  '/ 軸1目標位置
dst.Pos(2) = 300                   '/ 軸2目標位置
dst.Center(1) = 600                '/ 軸1中心位置
dst.Center(2) = 0                  '/ 軸2中心位置
dst.Direction(0) = MNET_DIR_CW     '/ 回転方向:CW
dst.StartMode = MNET_STAMODE_UD    '/ スタートモード:加減速

spd.SpdL(0) = 2000                 '/ FL速度設定
spd.SpdH(0) = 10000                '/ FH速度設定
spd.Acc(0) = 1.5                   '/ 加速データ
spd.Dec(0) = 3.0                   '/ 減速データ
spd.AdcSpc(0) = MNET_ADC_BY_TIME   '/ 加減速指定方法: 時間(秒)

MNET6045Arc 0, 0, &H6, dst, spd
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
Dim spd As New TMNET6045Spd          '/ 速度データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化
spd.Initialize()                     '/ 構造体初期化

dst.Pos(1) = 1000                    '/ 軸1目標位置
dst.Pos(2) = 300                     '/ 軸2目標位置
dst.Center(1) = 600                  '/ 軸1中心位置
dst.Center(2) = 0                    '/ 軸2中心位置
dst.Direction(0) = MNET_DIR_CW       '/ 回転方向:CW
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速

spd.SpdL(0) = 2000                   '/ FL速度設定
spd.SpdH(0) = 10000                  '/ FH速度設定
spd.Acc(0) = 1.5                     '/ 加速データ
spd.Dec(0) = 3.0                     '/ 減速データ
spd.AdcSpc(0) = MNET_ADC_BY_TIME     '/ 加減速指定方法: 時間(秒)

MNET6045Arc(0, 0, &H6, dst, spd)
```

ライン0、デバイス0、軸パターン6h(軸1+軸2)のAbs指定円弧補間移動を行います。

Motionnet®
Remote I/O & Remote Motion

8－1－1－8. MNET6045Continue

機能

連続移動

- 連続移動とはJog動作のような移動量指定の無い、指定方向へ動き続ける動作です。
- 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。

書式

[Visual C++]

```
short __stdcall MNET6045Continue(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045Continue Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045Continue Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン*** (8-2-1.軸パターン参照)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};              // 移動データ構造体定義,初期化

dst.Direction[0] = MNET_DIR_PLUS;    // 軸 0 動作方向
dst.Direction[2] = MNET_DIR_MINUS;   // 軸 2 動作方向
dst.StartMode = MNET_STAMODE_UD;     // スタートモード:加減速

MNET6045Continue(0, 0, 0x5, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst              '/ 移動データ構造体定義,初期化

dst.Direction(0) = MNET_DIR_PLUS     '/ 軸0動作方向
dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速

MNET6045Continue 0, 0, &H5, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化

dst.Direction(0) = MNET_DIR_PLUS     '/ 軸0動作方向
dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速

MNET6045Continue(0, 0, &H5, dst)
```

ライン0、デバイス0、軸パターン5h(軸0+軸2)の連続移動を行います。

8－1－1－9. MNET6045Org

機能

原点復帰移動起動

- 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。
- 原点復帰モードはMNET6045SetPar関数で設定します。

書式

[Visual C++]

```
short __stdcall MNET6045Org(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045Org Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045Org Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axPtn	***軸パターン***（8-2-1.軸パターン参照）
dst	***移動データ構造体***（8-2-5.移動データ構造体参照）

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};              // 移動データ構造体定義,初期化

dst.Direction[0] = MNET_DIR_PLUS;    // 軸0動作方向
dst.Direction[2] = MNET_DIR_MINUS;   // 軸2動作方向
dst.StartMode = MNET_STAMODE_FL;     // スタートモード:FL定速

MNET6045Org(0, 0, 0x5, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst              '/ 移動データ構造体定義,初期化

dst.Direction(0) = MNET_DIR_PLUS     '/ 軸0動作方向
dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_FL      '/ スタートモード:FL定速

MNET6045Org 0, 0, &H5, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化

dst.Direction(0) = MNET_DIR_PLUS     '/ 軸0動作方向
dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_FL      '/ スタートモード:FL定速

MNET6045Org(0, 0, &H5, dst)
```

ライン0、デバイス0、軸パターン5h(軸0+軸2)の原点復帰移動を行います。

8－1－1－10. MNET6045EscOrg

機能

ORG脱出移動起動

- 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。
- ORG脱出移動はスタートモード:加減速ではご利用できません。

書式

[Visual C++]

```
short __stdcall MNET6045EscOrg(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045EscOrg Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045EscOrg Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axPtn	***軸パターン***（8-2-1.軸パターン参照）
dst	***移動データ構造体***（8-2-5.移動データ構造体参照）

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};              // 移動データ構造体定義,初期化

dst.Direction[0] = MNET_DIR_PLUS;    // 軸0動作方向
dst.Direction[2] = MNET_DIR_MINUS;   // 軸2動作方向
dst.StartMode = MNET_STAMODE_FL;     // スタートモード:FL定速

MNET6045EscOrg(0, 0, 0x5, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst              '/ 移動データ構造体定義,初期化

dst.Direction(0) = MNET_DIR_PLUS     '/ 軸0動作方向
dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_FL      '/ スタートモード:FL定速

MNET6045EscOrg 0, 0, &H5, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化

dst.Direction(0) = MNET_DIR_PLUS     '/ 軸0動作方向
dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_FL      '/ スタートモード:FL定速

MNET6045EscOrg(0, 0, &H5, dst)
```

ライン0、デバイス0、軸パターン5h(軸0+軸2)のORG脱出移動を行います。

8－1－1－11. MNET6045SearchOrg

機能

原点サーチ移動起動

- 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。
- 原点抜け出し移動量は移動データ構造体のメンバー「Pos」の対応した軸のデータに設定します。
- 原点復帰モードはMNET6045SetPar関数で設定します。
- 原点サーチ動作は減速時間 ＜ 加速時間×2の速度設定で行ってください。

書式

[Visual C++]

```
short __stdcall MNET6045SearchOrg(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045SearchOrg Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045SearchOrg Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axPtn	***軸パターン***（8-2-1.軸パターン参照）
dst	***移動データ構造体***（8-2-5.移動データ構造体参照）

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};              // 移動データ構造体定義,初期化

dst.Direction[0] = MNET_DIR_PLUS;    // 軸0動作方向
dst.Direction[2] = MNET_DIR_MINUS;   // 軸2動作方向
dst.Pos[0] = 100;                    // 軸0原点抜出移動量
dst.Pos[2] = 50;                     // 軸2原点抜出移動量
dst.StartMode = MNET_STAMODE_UD;     // スタートモード:加減速

MNET6045SearchOrg(0, 0, 0x5, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst              '/ 移動データ構造体定義,初期化

dst.Direction(0) = MNET_DIR_PLUS     '/ 軸0動作方向
dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.Pos(0) = 100                     '/ 軸0原点抜出移動量
dst.Pos(2) = 50                      '/ 軸2原点抜出移動量
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速

MNET6045SearchOrg 0, 0, &H5, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化

dst.Direction(0) = MNET_DIR_PLUS     '/ 軸0動作方向
dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.Pos(0) = 100                     '/ 軸0原点抜出移動量
dst.Pos(2) = 50                      '/ 軸2原点抜出移動量
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速

MNET6045SearchOrg(0, 0, &H5, dst)
```

ライン0、デバイス0、軸パターン5h(軸0+軸2)の原点サーチ移動を行います。

8－1－1－12. MNET6045ToEL

機能

ELまでの移動を起動

・ 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。

書式

[Visual C++]

```
short __stdcall MNET6045ToEL(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045ToEL Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045ToEL Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン*** (8-2-1.軸パターン参照)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};              // 移動データ構造体定義,初期化

dst.Direction[0] = MNET_DIR_PLUS;    // 軸0動作方向
dst.Direction[2] = MNET_DIR_MINUS;   // 軸2動作方向
dst.StartMode = MNET_STAMODE_UD;     // スタートモード:加減速

MNET6045ToEL(0, 0, 0x5, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst              '/ 移動データ構造体定義,初期化

dst.Direction(0) = MNET_DIR_PLUS     '/ 軸0動作方向
dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速

MNET6045ToEL 0, 0, &H5, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化

dst.Direction(0) = MNET_DIR_PLUS     '/ 軸0動作方向
dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速

MNET6045ToEL(0, 0, &H5, dst)
```

ライン0、デバイス0、軸パターン5h(軸0+軸2)の「ELまで移動」を行います。

8－1－1－13. MNET6045EscEL

機能

EL脱出移動を起動

・ 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。

書式

[Visual C++]

```
short __stdcall MNET6045EscEL(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045EscEL Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045EscEL Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン*** (8-2-1.軸パターン参照)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};                 // 移動データ構造体定義,初期化

dst.Direction[0] = MNET_DIR_PLUS;       // 軸0動作方向
dst.Direction[2] = MNET_DIR_MINUS;      // 軸2動作方向
dst.StartMode = MNET_STAMODE_UD;        // スタートモード:加減速

MNET6045EscEL(0, 0, 0x5, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst                 '/ 移動データ構造体定義,初期化

dst.Direction(0) = MNET_DIR_PLUS        '/ 軸0動作方向
dst.Direction(2) = MNET_DIR_MINUS       '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_UD         '/ スタートモード:加減速

MNET6045EscEL 0, 0, &H5, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst             '/ 移動データ構造体定義,初期化
dst.Initialize()                        '/ 構造体初期化

dst.Direction(0) = MNET_DIR_PLUS        '/ 軸0動作方向
dst.Direction(2) = MNET_DIR_MINUS       '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_UD         '/ スタートモード:加減速

MNET6045EscEL(0, 0, &H5, dst)
```

ライン0、デバイス0、軸パターン5h(軸0+軸2)のEL脱出移動を行います。

8－1－1－14. MNET6045Ez

機能

EZ信号カウント動作を起動します。
(スタートコマンドを発行します)
指定した回数のEZ信号をカウントするまでの動作となります。

書式

[Visual C++]

```
short __stdcall MNET6045Ez(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045Ez Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045Ez Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン*** (8-2-1.軸パターン参照)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)

備考

EZカウント回数は事前にMNET6045SetPar関数で設定した値が適用されます。
加減速動作の場合はEZカウント完了により減速停止となります。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};                 // 移動データ構造体定義,初期化

dst.Direction[0] = MNET_DIR_PLUS;       // 軸0動作方向
dst.Direction[2] = MNET_DIR_MINUS;      // 軸2動作方向
dst.StartMode = MNET_STAMODE_FL;        // スタートモード:FL定速

MNET6045Ez(0, 0, 0x5, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst                 '/ 移動データ構造体定義,初期化

dst.Direction(0) = MNET_DIR_PLUS        '/ 軸0動作方向
dst.Direction(2) = MNET_DIR_MINUS       '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_FL         '/ スタートモード:FL定速

MNET6045Ez 0, 0, &H5, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst             '/ 移動データ構造体定義,初期化
dst.Initialize()                        '/ 構造体初期化

dst.Direction(0) = MNET_DIR_PLUS        '/ 軸0動作方向
dst.Direction(2) = MNET_DIR_MINUS       '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_FL         '/ スタートモード:FL定速

MNET6045Ez(0, 0, &H5, dst)
```

ライン0、デバイス0、軸パターン5h(軸0+軸2)のEZカウント移動を行います。

8－1－1－15. MNET6045Stop

機能

軸を停止します。

・ 即停止、減速停止などの停止方法の選択は引数で行います。

書式

[Visual C++]

```
short __stdcall MNET6045Stop(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    short stopMode);
```

[Visual Basic]

```
Declare Function MNET6045Stop Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByVal stopMode As Integer)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045Stop Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByVal stopMode As Short)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン*** (8-2-1.軸パターン参照)
stopMode	***停止モード*** (8-3-3-7.停止モード参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
MNET6045Stop(0, 0, 0x3, MNET_SDSTOP);
```

[Visual Basic]

```
MNET6045Stop 0, 0, &H3, MNET_SDSTOP
```

[VB.NET]

```
MNET6045Stop(0, 0, &H3, MNET_SDSTOP)
```

ライン0、デバイス0,軸パターン3h(軸0+軸1)を減速停止させます。

8－1－1－16. MNET6045SetSpeedNo

機能

（非補間動作）速度データ設定（軸番号指定）

- 本関数により設定された速度データは以降の非補間動作の動作速度として適用されます。
- 補間動作実行により、該当軸の速度データは無効となりますので、非補間動作実行前には改めて速度データの設定が必要となります。
- 速度データは速度データ構造体の対象の軸番号の要素に設定します。

書式

[Visual C++]

```
short __stdcall MNET6045SetSpeedNo(
                    short lineNo,
                    short devNo,
                    short axNo,
                    TMNET6045Spd* spd,
                    short startMode);
```

[Visual Basic]

```
Declare Function MNET6045SetSpeedNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByRef spd As TMNET6045Spd,
                    ByVal startMode As Integer)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045SetSpeedNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByRef spd As TMNET6045Spd,
                    ByVal startMode As Short)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axNo	***軸番号***（0始まり）
spd	***速度データ構造体***（8-2-4.速度データ構造体参照）
startMode	***スタートモード***（8-2-3.スタートモード参照）

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

[例1]

```
TMNET6045Spd spd = {0};              // 速度データ構造体定義,初期化

spd.SpdL[2] = 2000;                  // FL速度設定
spd.SpdH[2] = 10000;                 // FH速度設定
spd.Acc[2] = 1.5;                    // 加速データ
spd.Dec[2] = 5;                      // 減速データ
spd.AdcSpc[2] = MNET_ADC_BY_TIME;    // 加減速指定方法: 時間(秒)

MNET6045SetSpeedNo(0, 0, 2, &spd, MNET_STAMODE_UD);
```

ライン0、デバイス0、軸2の速度データを設定します。

[例2]

```
TMNET6045Spd spd = {0};              // 速度データ構造体定義,初期化

spd.SpdL[1] = 2000;                  // FL速度設定
spd.SpdH[1] = 10000;                 // FH速度設定
spd.Acc[1] = 1000;                   // 加速データ
spd.Dec[1] = 2000;                   // 減速データ
spd.AdcSpc[1] = MNET_ADC_BY_RATE;    // 加減速指定方法: レート(秒あたりの増減分パルス)
spd.SMode[1] = MNET_USE_S;           // S字使用

MNET6045SetSpeedNo(0, 0, 1, &spd, MNET_STAMODE_UD);
```

ライン0、デバイス0、軸1の速度データを設定します。

[Visual Basic]

［例1］

```
Dim spd As TMNET6045Spd              '/ 速度データ構造体定義,初期化

spd.SpdL(2) = 2000                   '/ FL速度設定
spd.SpdH(2) = 10000                  '/ FH速度設定
spd.Acc(2) = 1.5                     '/ 加速データ
spd.Dec(2) = 5                       '/ 減速データ
spd.AdcSpc(2) = MNET_ADC_BY_TIME     '/ 加減速指定方法: 時間(秒)

MNET6045SetSpeedNo 0, 0, 2, spd, MNET_STAMODE_UD
```

ライン0、デバイス0、軸2の速度データを設定します。

［例2］

```
Dim spd As TMNET6045Spd              '/ 速度データ構造体定義,初期化

spd.SpdL(1) = 2000                   '/ FL速度設定
spd.SpdH(1) = 10000                  '/ FH速度設定
spd.Acc(1) = 1000                    '/ 加速データ
spd.Dec(1) = 2000                    '/ 減速データ
spd.AdcSpc(1) = MNET_ADC_BY_RATE     '/ 加減速指定方法: レート(秒あたりの増減分パルス)
spd.SMode(1) = MNET_USE_S            '/ S字使用

MNET6045SetSpeedNo 0, 0, 1, spd, MNET_STAMODE_UD
```

ライン0、デバイス0、軸1の速度データを設定します。

[VB.NET]

[例1]

```
Dim spd As New TMNET6045Spd          '/ 速度データ構造体定義,初期化
spd.Initialize()                     '/ 構造体初期化

spd.SpdL(2) = 2000                   '/ FL速度設定
spd.SpdH(2) = 10000                  '/ FH速度設定
spd.Acc(2) = 1.5                     '/ 加速データ
spd.Dec(2) = 5                       '/ 減速データ
spd.AdcSpc(2) = MNET_ADC_BY_TIME     '/ 加減速指定方法: 時間(秒)

MNET6045SetSpeedNo(0, 0, 2, spd, MNET_STAMODE_UD)
```

ライン0、デバイス0、軸2の速度データを設定します。

[例2]

```
Dim spd As New TMNET6045Spd          '/ 速度データ構造体定義,初期化
spd.Initialize()                     '/ 構造体初期化

spd.SpdL(1) = 2000                   '/ FL速度設定
spd.SpdH(1) = 10000                  '/ FH速度設定
spd.Acc(1) = 1000                    '/ 加速データ
spd.Dec(1) = 2000                    '/ 減速データ
spd.AdcSpc(1) = MNET_ADC_BY_RATE     '/ 加減速指定方法: レート(秒あたりの増減分パルス)
spd.SMode(1) = MNET_USE_S            '/ S字使用

MNET6045SetSpeedNo(0, 0, 1, spd, MNET_STAMODE_UD)
```

ライン0、デバイス0、軸1の速度データを設定します。

8－1－1－17. MNET6045PtpIncNo

機能

Ptp移動(Inc指定)（軸番号指定）

- 相対移動量指定のPtp移動をスタートします。
- 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。

書式

[Visual C++]

```
short __stdcall MNET6045PtpIncNo(
                    short lineNo,
                    short devNo,
                    short axNo,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045PtpIncNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045PtpIncNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axNo	***軸番号***（0始まり）
dst	***移動データ構造体***（8-2-5.移動データ構造体参照）

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};                 // 移動データ構造体定義,初期化

dst.Pos[0] = 10000;                     // 軸0移動量
dst.StartMode = MNET_STAMODE_FL;        // スタートモード:FL定速

MNET6045PtpIncNo(0, 0, 0, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst                 '/ 移動データ構造体定義,初期化

dst.Pos(0) = 10000                      '/ 軸0移動量
dst.StartMode = MNET_STAMODE_FL         '/ スタートモード:FL定速

MNET6045PtpIncNo 0, 0, 0, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst             '/ 移動データ構造体定義,初期化
dst.Initialize()                        '/ 構造体初期化

dst.Pos(0) = 10000                      '/ 軸0移動量
dst.StartMode = MNET_STAMODE_FL         '/ スタートモード:FL定速

MNET6045PtpIncNo 0, 0, 0, dst
```

ライン0、デバイス0、軸0のPtp(Inc)移動を行います。

8－1－1－18. MNET6045PtpNo

機能

Ptp移動(Abs指定)（軸番号指定）

- 絶対移動量指定のPtp移動をスタートします。
- 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。

書式

[Visual C++]

```
short __stdcall MNET6045PtpNo(
                    short lineNo,
                    short devNo,
                    short axNo,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045PtpNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045PtpNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axNo	***軸番号***（0始まり）
dst	***移動データ構造体***（8-2-5.移動データ構造体参照）

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};              // 移動データ構造体定義,初期化

dst.Pos[2] = -6000;                  // 軸2目標位置
dst.StartMode = MNET_STAMODE_UD;     // スタートモード:加減速

MNET6045PtpNo(0, 0, 2, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst              '/ 移動データ構造体定義,初期化

dst.Pos(2) = -6000                   '/ 軸2目標位置
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速

MNET6045PtpNo 0, 0, 2, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化

dst.Pos(2) = -6000                   '/ 軸2目標位置
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速

MNET6045PtpNo(0, 0, 2, dst)
```

ライン0、デバイス0、軸2のPtp(Abs)移動を行います。

8－1－1－19. MNET6045ContinueNo

機能

連続移動 （軸番号指定）

- 連続移動とはJog動作のような移動量指定の無い、指定方向へ動き続ける動作です。
- 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。

書式

[Visual C++]

```
short __stdcall MNET6045ContinueNo(
                    short lineNo,
                    short devNo,
                    short axNo,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045ContinueNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045ContinueNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};                 // 移動データ構造体定義,初期化

dst.Direction[0] = MNET_DIR_PLUS;       // 軸0動作方向
dst.StartMode = MNET_STAMODE_UD;        // スタートモード:加減速

MNET6045ContinueNo(0, 0, 0, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst                 '/ 移動データ構造体定義,初期化

dst.Direction(0) = MNET_DIR_PLUS        '/ 軸0動作方向
dst.StartMode = MNET_STAMODE_UD         '/ スタートモード:加減速

MNET6045ContinueNo 0, 0, 0, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst             '/ 移動データ構造体定義,初期化
dst.Initialize()                        '/ 構造体初期化

dst.Direction(0) = MNET_DIR_PLUS        '/ 軸0動作方向
dst.StartMode = MNET_STAMODE_UD         '/ スタートモード:加減速

MNET6045ContinueNo(0, 0, 0, dst)
```

ライン0、デバイス0、軸0の連続移動を行います。

Motionnet®
Remote I/O & Remote Motion

8－1－1－20. MNET6045OrgNo

機能

原点復帰移動起動 (軸番号指定)

- 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。
- 原点復帰モードはMNET6045SetPar関数で設定します。

書式

[Visual C++]

```
short __stdcall MNET6045OrgNo(
                    short lineNo,
                    short devNo,
                    short axNo,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045OrgNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045OrgNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};              // 移動データ構造体定義,初期化

dst.Direction[2] = MNET_DIR_MINUS;   // 軸2動作方向
dst.StartMode = MNET_STAMODE_FL;     // スタートモード:FL定速

MNET6045OrgNo(0, 0, 2, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst              '/ 移動データ構造体定義,初期化

dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_FL      '/ スタートモード:FL定速

MNET6045OrgNo 0, 0, 2, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化

dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_FL      '/ スタートモード:FL定速

MNET6045OrgNo(0, 0, 2, dst)
```

ライン0、デバイス0、軸2の原点復帰移動を行います。

8－1－1－21. MNET6045EscOrgNo

機能

ORG脱出移動起動 （軸番号指定）

- 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。
- ORG脱出移動はスタートモード:加減速ではご利用できません。

書式

[Visual C++]

```
short __stdcall MNET6045EscOrgNo(
                    short lineNo,
                    short devNo,
                    short axNo,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045EscOrgNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045EscOrgNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};                 // 移動データ構造体定義,初期化

dst.Direction[0] = MNET_DIR_PLUS;       // 軸0動作方向
dst.StartMode = MNET_STAMODE_FL;        // スタートモード:FL定速

MNET6045EscOrgNo(0, 0, 0, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst                 '/ 移動データ構造体定義,初期化

dst.Direction(0) = MNET_DIR_PLUS        '/ 軸0動作方向
dst.StartMode = MNET_STAMODE_FL         '/ スタートモード:FL定速

MNET6045EscOrgNo 0, 0, 0, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst             '/ 移動データ構造体定義,初期化
dst.Initialize()                        '/ 構造体初期化

dst.Direction(0) = MNET_DIR_PLUS        '/ 軸0動作方向
dst.StartMode = MNET_STAMODE_FL         '/ スタートモード:FL定速

MNET6045EscOrgNo(0, 0, 0, dst)
```

ライン0、デバイス0、軸0のORG脱出移動を行います。

8－1－1－22. MNET6045SearchOrgNo

機能

原点サーチ移動起動 （軸番号指定）

- 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。
- 原点抜け出し移動量は移動データ構造体のメンバー「Pos」の対応した軸のデータに設定します。
- 原点復帰モードはMNET6045SetPar関数で設定します。
- 原点サーチ動作は減速時間 ＜ 加速時間×2の速度設定で行ってください。

書式

[Visual C++]

```
short __stdcall MNET6045SearchOrgNo
                    short lineNo,
                    short devNo,
                    short axNo,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045SearchOrgNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045SearchOrgNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};                 // 移動データ構造体定義,初期化

dst.Direction[0] = MNET_DIR_PLUS;       // 軸0動作方向
dst.Pos[0] = 100;                       // 軸0原点抜出移動量
dst.StartMode = MNET_STAMODE_UD;        // スタートモード:加減速

MNET6045SearchOrgNo(0, 0, 0, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst                 '/ 移動データ構造体定義,初期化

dst.Direction(0) = MNET_DIR_PLUS        '/ 軸0動作方向
dst.Pos(0) = 100                        '/ 軸0原点抜出移動量
dst.StartMode = MNET_STAMODE_UD         '/ スタートモード:加減速

MNET6045SearchOrgNo 0, 0, 0, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst             '/ 移動データ構造体定義,初期化
dst.Initialize()                        '/ 軸0動作方向
dst.Pos(0) = 100                        '/ 軸0原点抜出移動量
dst.StartMode = MNET_STAMODE_UD         '/ スタートモード:加減速

MNET6045SearchOrgNo(0, 0, 0, dst)
```

ライン0、デバイス0、軸0の原点サーチ移動を行います。

8－1－1－23. MNET6045ToELNo

機能

ELまでの移動を起動 （軸番号指定）

・ 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。

書式

[Visual C++]

```
short __stdcall MNET6045ToELNo(
                    short lineNo,
                    short devNo,
                    short axNo,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045ToELNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045ToELNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};              // 移動データ構造体定義,初期化

dst.Direction[2] = MNET_DIR_MINUS;   // 軸2動作方向
dst.StartMode = MNET_STAMODE_UD;     // スタートモード:加減速

MNET6045ToELNo(0, 0, 2, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst              '/ 移動データ構造体定義,初期化

dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速

MNET6045ToELNo 0, 0, 2, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化

dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速

MNET6045ToELNo(0, 0, 2, dst)
```

ライン0、デバイス0、軸2の「ELまで移動」を行います。

8－1－1－24. MNET6045EscELNo

機能

EL脱出移動を起動 （軸番号指定）

・ 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。

書式

[Visual C++]

```
short __stdcall MNET6045EscELNo(
                    short lineNo,
                    short devNo,
                    short axNo,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045EscELNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045EscELNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};              // 移動データ構造体定義,初期化

dst.Direction[2] = MNET_DIR_MINUS;   // 軸2動作方向
dst.StartMode = MNET_STAMODE_UD;     // スタートモード:加減速

MNET6045EscELNo(0, 0, 2, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst              '/ 移動データ構造体定義,初期化

dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速

MNET6045EscELNo 0, 0, 2, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化

dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_UD      '/ スタートモード:加減速

MNET6045EscELNo(0, 0, 2, dst)
```

ライン0、デバイス0、軸2のEL脱出移動を行います。

8－1－1－25. MNET6045EzNo

機能

EZ信号カウント動作を起動します。
(スタートコマンドを発行します)
指定した回数のEZ信号をカウントするまでの動作となります。

書式

[Visual C++]

```
short __stdcall MNET6045EzNo(
                    short lineNo,
                    short devNo,
                    short axNo,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045EzNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045EzNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)

備考

EZカウント回数は事前にMNET6045SetPar関数で設定した値が適用されます。
加減速動作の場合はEZカウント完了により減速停止となります。

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};              // 移動データ構造体定義,初期化

dst.Direction[2] = MNET_DIR_MINUS;   // 軸2動作方向
dst.StartMode = MNET_STAMODE_FL;     // スタートモード:FL定速

MNET6045EzNo(0, 0, 2, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst              '/ 移動データ構造体定義,初期化

dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_FL      '/ スタートモード:FL定速

MNET6045EzNo 0, 0, 2, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化

dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_FL      '/ スタートモード:FL定速

MNET6045EzNo(0, 0, 2, dst)
```

ライン0、デバイス0、軸2のEZカウント移動を行います。

8－1－1－26. MNET6045StopNo

機能

軸を停止します。（軸番号指定）

・ 即停止、減速停止などの停止方法の選択は引数で行います。

書式

[Visual C++]

```
short __stdcall MNET6045StopNo(
                    short lineNo,
                    short devNo,
                    short axNo,
                    short stopMode);
```

[Visual Basic]

```
Declare Function MNET6045StopNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByVal stopMode As Integer)
                      As Integer
```

[VB.NET]

```
Declare Function MNET6045StopNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByVal stopMode As Short)
                      As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axNo	***軸番号***（0始まり）
stopMode	***停止モード***（8-3-3-7.停止モード参照）

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

```
MNET6045StopNo(0, 0, 1, MNET_SDSTOP);
```

[Visual Basic]

```
MNET6045StopNo 0, 0, 1, MNET_SDSTOP
```

[VB.NET]

```
MNET6045StopNo(0, 0, 1, MNET_SDSTOP)
```

ライン0、デバイス0,軸1を減速停止させます。

8－1－1－27. MNET6045GetBitStruct

機能

ビットデータ読出し(構造体に一括)

書式

[Visual C++]

```
short __stdcall MNET6045GetBitStruct(
                    short lineNo,
                    short devNo,
                    short axNo,
                    TMNET6045BitData* bit);
```

[Visual Basic]

```
Declare Function MNET6045GetBitStruct Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByRef bit As TMNET6045BitData)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045GetBitStruct Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByRef bit As TMNET6045BitData)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
bit	***ビットデータ構造体*** (8-3-1-1.ビットデータ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045BitData bit[4] = {0};        // ビットデータ構造体定義,初期化

MNET6045GetBitStruct(0, 0, 1, &bit[1]);
```

[Visual Basic]

```
Dim bit(0 to 3) As TMNET6045BitData   '/ ビットデータ構造体定義,初期化

MNET6045GetBitStruct 0, 0, 1, bit(1)
```

[VB.NET]

```
Dim bit(0 to 3) As TMNET6045BitData   '/ ビットデータ構造体定義,初期化

MNET6045GetBitStruct(0, 0, 1, bit(1))
```

ライン0、デバイス0,軸1のビットデータを構造体に一括取得します。

8－1－1－28. MNET6045GetBit

機能

ビットデータ読出し

書式

[Visual C++]

```
short __stdcall MNET6045GetBit(
                    short lineNo,
                    short devNo,
                    short axNo,
                    short sel,
                    short* val);
```

[Visual Basic]

```
Declare Function MNET6045GetBit Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByVal sel As Integer,
                    ByRef val As Integer)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045GetBit Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByVal sel As Short,
                    ByRef val As Short)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
sel	***対象選択*** (8-3-3-9.ビット選択参照)
val	***取得データ格納***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
short val;   // 取得データ格納

MNET6045GetBit(0, 0, 1, MNET6045_BIT_MPOUT1, &val);
```

[Visual Basic]

```
Dim val As Integer   ‘/ 取得データ格納

MNET6045GetBit 0, 0, 1, MNET6045_BIT_MPOUT1, val
```

[VB.NET]

```
Dim val As Short   ‘/ 取得データ格納

MNET6045GetBit(0, 0, 1, MNET6045_BIT_MPOUT1, val)
```

ライン0、デバイス0,軸1のビットMPOUT1を取得します。

8－1－1－29. MNET6045SetBit

機能

ビットデータ設定

書式

[Visual C++]

```
short __stdcall MNET6045SetBit(
                    short lineNo,
                    short devNo,
                    short axNo,
                    short sel,
                    short val);
```

[Visual Basic]

```
Declare Function MNET6045SetBit Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByVal sel As Integer,
                    ByVal val As Integer)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045SetBit Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByVal sel As Short,
                    ByVal val As Short)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
sel	***対象選択*** (8-3-3-9.ビット選択参照)
val	***設定値***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
MNET6045SetBit(0, 0, 1, MNET6045_BIT_MPOUT1, 1);
```

[Visual Basic]

```
MNET6045SetBit 0, 0, 1, MNET6045_BIT_MPOUT1, 1
```

[VB.NET]

```
MNET6045SetBit(0, 0, 1, MNET6045_BIT_MPOUT1, 1)
```

ライン0、デバイス0,軸1のビットMPOUT1を1に設定します。

Motionnet® Remote I/O & Remote Motion

8－1－1－30. MNET6045GetSpeed

機能

現在速度取得

- 速度取得中にプリレジスタ状態が変化した場合はエラーが返されます。
- 指定した軸が直線補間または円弧補間に含まれる場合は、補間動作の合成速度が取得されます。

書式

[Visual C++]

```
short __stdcall MNET6045GetSpeed(
                    short lineNo,
                    short devNo,
                    short axNo,
                    double* val);
```

[Visual Basic]

```
Declare Function MNET6045GetSpeed Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByRef val As Double)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045GetSpeed Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByRef val As Double)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
val	***取得データ格納***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
double val;

MNET6045GetSpeed(0, 0, 1, &val);
```

[Visual Basic]

```
Dim val As Double

MNET6045GetSpeed 0, 0, 1, val
```

[VB.NET]

```
Dim val As Double

MNET6045GetSpeed(0, 0, 1, val)
```

ライン0、デバイス0,軸1の速度データを取得します。

8－1－1－31. MNET6045GetCounter

機能

カウンタ値取得

書式

[Visual C++]

```
short __stdcall MNET6045GetCounter(
                    short lineNo,
                    short devNo,
                    short axNo,
                    short sel,
                    long* val);
```

[Visual Basic]

```
Declare Function MNET6045GetCounter Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByVal sel As Integer,
                    ByRef val As Long)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045GetCounter Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByVal sel As Short,
                    ByRef val As Integer)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
sel	***対象選択*** (8-3-3-5.カウンタ指定参照)
val	***取得データ格納***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
long val;

MNET6045GetCounter(0, 0, 1, MNET_SEL_COUNTER1, &val);
```

[Visual Basic]

```
Dim val As Long

MNET6045GetCounter 0, 0, 1, MNET_SEL_COUNTER1, val
```

[VB.NET]

```
Dim val As Integer

MNET6045GetCounter(0, 0, 1, MNET_SEL_COUNTER1, val)
```

ライン0、デバイス0,軸1のカウンタ1(指令位置カウンタ)の値を取得します。

8－1－1－32. MNET6045SetCounter

機能

カウンタ値設定

書式

[Visual C++]

```
short __stdcall MNET6045SetCounter(
                    short lineNo,
                    short devNo,
                    short axNo,
                    short sel,
                    long val);
```

[Visual Basic]

```
Declare Function MNET6045SetCounter Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByVal sel As Integer,
                    ByVal val As Long)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045SetCounter Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByVal sel As Short,
                    ByVal val As Integer)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
sel	***対象選択*** (8-3-3-5.カウンタ指定参照)
val	***設定値***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
MNET6045SetCounter(0, 0, 1, MNET_SEL_COUNTER2, 0);
```

[Visual Basic]

```
MNET6045SetCounter 0, 0, 1, MNET_SEL_COUNTER2, 0
```

[VB.NET]

```
MNET6045SetCounter(0, 0, 1, MNET_SEL_COUNTER2, 0)
```

ライン0、デバイス0,軸1のカウンタ2(フィードバックカウンタ)に0を設定します。

8－1－1－33. MNET6045Reset

機能

LSIリセット

・ MNET6045SetPar関数などで設定したパラメータデータなども全てリセットされます。

書式

[Visual C++]

```
short __stdcall MNET6045Reset(
                    short lineNo,
                    short devNo)
```

[Visual Basic]

```
Declare Function MNET6045Reset Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045Reset Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

```
MNET6045Reset(0, 0);
```

[Visual Basic]

```
MNET6045Reset 0, 0
```

[VB.NET]

```
MNET6045Reset(0, 0)
```

ライン0、デバイス0のLSIをリセットします。

8－1－1－34. MNET6045CancelPreReg

機能

次動作キャンセル

・ 次動作については「8-2-22.次動作設定」を参照ください。

書式

[Visual C++]

```
short __stdcall MNET6045CancelPreReg(
                    short lineNo,
                    short devNo,
                    short axPtn);
```

[Visual Basic]

```
Declare Function MNET6045CancelPreReg Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045CancelPreReg Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン*** (8-2-1.軸パターン参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
MNET6045CancelPreReg(0, 0, 0x3);
```

[Visual Basic]

```
MNET6045CancelPreReg 0, 0, &H3
```

[VB.NET]

```
MNET6045CancelPreReg(0, 0, &H3)
```

ライン0、デバイス0,軸パターン3h(軸0+軸1)の次動作をキャンセルします。

8－1－1－35. MNET6045StopChk

機能

軸停止確認

- 指定した軸パターンの軸の動作状態が取得できます。
- 軸パターンに含まれる軸番号の小さい軸から状態が確認されます。
- 異常停止軸(エラー割込要因にデータがある軸)が見つかった場合、その時点で確認は中断されます。
- 同時スタート(CSTA)信号他の待ち状態は、動作中として扱われます。

書式

[Visual C++]

```
short __stdcall MNET6045StopChk(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    short* stopSts);
```

[Visual Basic]

```
Declare Function MNET6045StopChk Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef stopSts As Integer)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045StopChk Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef stopSts As Short)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン*** (8-2-1.軸パターン参照)
stopSts	***状態取得***

0 : 動作中軸あり
1 : 全軸正常停止
-1 : 異常停止軸有り
-2 : その他のエラー(指定値不正など)により状態取得失敗

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

Motionnet® Remote I/O & Remote Motion

使用例

[Visual C++]

```
short stopSts;

MNET6045StopChk(0, 0, 0x3, &stopSts);
```

[Visual Basic]

```
Dim stopSts As Integer

MNET6045StopChk 0, 0, &H3, stopSts
```

[VB.NET]

```
Dim stopSts As Short

MNET6045StopChk(0, 0, &H3, stopSts)
```

ライン0、デバイス0,軸パターン3h(軸0+軸1)の停止状態を確認します。

8－1－1－36. MNET6045SetDLLPar

機能

DLLパラメータ設定
DLLパラメータとはDLLの動作を設定するパラメータです。

書式

[Visual C++]

```
short __stdcall MNET6045SetDLLPar(
                    short lineNo,
                    short devNo,
                    short axNo,
                    short itemNo,
                    short val);
```

[Visual Basic]

```
Declare Function MNET6045SetDLLPar Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByVal itemNo As Integer,
                    ByVal val As Integer)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045SetDLLPar Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByVal itemNo As Short,
                    ByVal val As Short)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
itemNo	***項目選択*** (8-3-3-11.DLL内部パラメータ項目番号定義参照)
val	***設定値***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
MNET6045SetDLLPar(0, 0, 0, MNET6045_DLLPAR_ENDLESS_ARC, 1);
```

[Visual Basic]

```
MNET6045SetDLLPar 0, 0, 0, MNET6045_DLLPAR_ENDLESS_ARC, 1
```

[VB.NET]

```
MNET6045SetDLLPar(0, 0, 0, MNET6045_DLLPAR_ENDLESS_ARC, 1)
```

ライン0、デバイス0のDLLパラメータ「無限円弧指令時処理」に’1’（エラーとする）を設定します。
（この項目の場合はボード単位での指定となっておりますのでaxNoの値は無視されます。）

8－1－1－37. MNET6045SetPar

機能

パラメータ設定

書式

[Visual C++]

```
short __stdcall MNET6045SetPar(
                    short lineNo,
                    short devNo,
                    short axNo,
                    short itemNo,
                    long val);
```

[Visual Basic]

```
Declare Function MNET6045SetPar Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByVal itemNo As Integer,
                    ByVal val As Long)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045SetPar Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByVal itemNo As Short,
                    ByVal val As Integer)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
itemNo	***項目選択*** (8-3-3-10.パラメータ項目番号定義参照)
val	***設定値***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
MNET6045SetPar(0, 0, 0, MNET6045_PAR_ORG_MODE, 2);
```

[Visual Basic]

```
MNET6045SetPar 0, 0, 0, MNET6045_PAR_ORG_MODE, 2
```

[VB.NET]

```
MNET6045SetPar(0, 0, 0, MNET6045_PAR_ORG_MODE, 2)
```

ライン0、デバイス0、軸0のパラメータ「原点復帰モードに」に’2’を設定します。

8－1－1－38. MNET6045GetPar

機能

パラメータ値取得
項目についてはパラメータ識別子定義の項を参照ください。

書式

[Visual C++]

```
short __stdcall MNET6045GetPar(
                    short lineNo,
                    short devNo,
                    short axNo,
                    short itemNo,
                    long* val);
```

[Visual Basic]

```
Declare Function MNET6045GetPar Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByVal itemNo As Integer,
                    ByRef val As Long)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045GetPar Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByVal itemNo As Short,
                    ByRef val As Integer)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
itemNo	***項目選択*** (8-3-3-10.パラメータ項目番号定義参照)
val	***取得値格納***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
long val;

MNET6045GetPar(0, 0, 0, MNET6045_PAR_ALM_LOGIC, &val);
```

[Visual Basic]

```
Dim val As Long

MNET6045GetPar 0, 0, 0, MNET6045
```

[VB.NET]

```
Dim val As Integer

MNET6045GetPar(0, 0, 0, MNET6045_PAR_ALM_LOGIC, val)
```

ライン0、デバイス0、軸0のパラメータ「ALM信号論理」のデータを取得します。

8－1－1－39. MNET6045FixSpeedRange

機能

速度レンジを固定します。
各レンジの範囲については識別子定義の項を参照ください。

書式

[Visual C++]

```
short __stdcall MNET6045FixSpeedRange(
                    short lineNo,
                    short devNo,
                    short axNo,
                    short sel);
```

[Visual Basic]

```
Declare Function MNET6045FixSpeedRange Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByVal sel As Integer)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045FixSpeedRange Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByVal sel As Short)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axNo	***軸番号***（0始まり）
sel	***レンジ選択***（8-3-3-8.速度倍率レンジ参照）

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

```
MNET6045FixSpeedRange(0, 0, 0, MNET6045_SPD_100_RANGE);
```

[Visual Basic]

```
MNET6045FixSpeedRange 0, 0, 0, MNET6045_SPD_100_RANGE
```

[VB.NET]

```
MNET6045FixSpeedRange(0, 0, 0, MNET6045_SPD_100_RANGE)
```

ライン0、デバイス0、軸0の速度レンジを「100倍レンジ」に固定します。

8－1－1－40. MNET6045UnFixSpeedRange

機能

速度レンジ固定解除

書式

[Visual C++]

```
short __stdcall MNET6045UnFixSpeedRange(
                    short lineNo,
                    short devNo,
                    short axNo);
```

[Visual Basic]

```
Declare Function MNET6045UnFixSpeedRange Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045UnFixSpeedRange Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
MNET6045UnFixSpeedRange(0, 0, 0);
```

[Visual Basic]

```
MNET6045UnFixSpeedRange 0, 0, 0
```

[VB.NET]

```
MNET6045UnFixSpeedRange(0, 0, 0)
```

ライン0、デバイス0、軸0の速度レンジ固定を解除します。

8－1－1－41. MNET6045SetFASpeed

機能

補正速度設定

- 補正速度はバックラッシュ補正や一部の原点復帰モードで適用される速度です。
- 本関数ではDLL内部データの設定が行われます。以降のMNET6045SetSpeed関数実行により、計算が行われハードウェアへの設定が行われます。
- 設定しない場合は当該動作の速度倍率内最小速度が適用されます。

書式

[Visual C++]

```
short __stdcall MNET6045SetFASpeed(
                    short lineNo,
                    short devNo,
                    short axNo,
                    double spdA);
```

[Visual Basic]

```
Declare Function MNET6045SetFASpeed Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByVal spdA As Double)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045SetFASpeed Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByVal spdA As Double)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axNo	***軸番号***（0始まり）
spdA	***設定値***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

```
MNET6045SetFASpeed(0, 0, 0, 100);
```

[Visual Basic]

```
MNET6045SetFASpeed 0, 0, 0, 100
```

[VB.NET]

```
MNET6045SetFASpeed(0, 0, 0, 100)
```

ライン0、デバイス0、軸0の補正速度に「100(pps)」を設定します。

Motionnet® Remote I/O & Remote Motion

8－1－1－42. MNET60450vrideSpd

機能

速度オーバーライド

加減速スタートで起動した動作の速度を変更します。

- 残りの移動量で指定速度まで到達できない場合はエラーが返され、オーバーライドは行なわれません。
- 指定した速度が動作の速度レンジを超える場合はエラーとなります。
- 加減速はスタート時の加減速設定と同じ傾きとなります。
- 指定軸が直線補間動作中の場合、指定した速度は動作中の直線補間移動の合成速度となります。

書式

[Visual C++]

```
short __stdcall MNET60450vrideSpd(
                    short lineNo,
                    short devNo,
                    short axNo,
                    double newSpeed);
```

[Visual Basic]

```
Declare Function MNET60450vrideSpd Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByVal newSpeed As Double)
                          As Integer
```

[VB.NET]

```
Declare Function MNET60450vrideSpd Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByVal newSpeed As Double)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axNo	***軸番号***（0始まり）
newSpeed	***設定値***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

```
MNET6045OvrideSpeed(0, 0, 0, 10000);
```

[Visual Basic]

```
MNET6045OvrideSpeed 0, 0, 0, 10000
```

[VB.NET]

```
MNET6045OvrideSpeed(0, 0, 0, 10000)
```

ライン0、デバイス0、軸0の速度を「10000(pps)」にオーバーライドします。

8－1－1－43. MNET6045GetRemain

機能

移動残パルス数取得

- 取得中にプリレジスタ状態が変化した場合はエラーが返されます。
- 位置決め動作以外での取得データは無効です。
- 指定軸が円弧補間に含まれる場合は円弧補間としての残パルスが取得されます。
- 円弧補間以外の場合は指定軸の残パルスが取得されます。

書式

[Visual C++]

```
short __stdcall MNET6045GetRemain(
                    short lineNo,
                    short devNo,
                    short axNo,
                    long* val);
```

[Visual Basic]

```
Declare Function MNET6045GetRemain Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByRef val As Long)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045GetRemain Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Shor,
                    ByRef val As Integer)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
val	***取得データ格納***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
long val;

MNET6045GetRemain(0, 0, 1, &val);
```

[Visual Basic]

```
Dim val As Long

MNET6045GetRemain 0, 0, 1, val
```

[VB.NET]

```
Dim val As Integer

MNET6045GetRemain(0, 0, 1, val)
```

ライン0、デバイス0,軸1の移動残パルスを取得します。

8－1－1－44. MNET6045GetLtc

機能

ラッチデータ取得
ラッチタイミングなどはMNET6045SetPar関数で設定します。

書式

[Visual C++]

```
short __stdcall MNET6045GetLtc(
                    short lineNo,
                    short devNo,
                    short axNo,
                    short sel,
                    long* val);
```

[Visual Basic]

```
Declare Function MNET6045GetLtc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByVal sel As Integer,
                    ByRef val As Long)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045GetLtc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByVal sel As Short,
                    ByRef val As Integer)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
sel	***対象ラッチ選択***(8-3-3-6.ラッチ指定参照)
val	***取得データ格納***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。(8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
long val;

MNET6045GetLtc(0, 0, 1, MNET6045_SEL_LTC1, &val);
```

[Visual Basic]

```
Dim val As Long

MNET6045GetLtc 0, 0, 1, MNET6045_SEL_LTC1, val
```

[VB.NET]

```
Dim val As Integer

MNET6045GetLtc(0, 0, 1, MNET6045_SEL_LTC1, val)
```

ライン0、デバイス0,軸1のラッチ1(CNT1ラッチ)データを取得します。

8－1－1－45. MNET6045SoftLtc

機能

ラッチ入力代行コマンド発行

書式

[Visual C++]

```
short __stdcall MNET6045SoftLtc(
                    short lineNo,
                    short devNo,
                    short axPtn);
```

[Visual Basic]

```
Declare Function MNET6045SoftLtc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045SoftLtc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン*** (8-2-1.軸パターン参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
MNET6045SoftLtc(0, 0, 0x3);
```

[Visual Basic]

```
MNET6045SoftLtc 0, 0, &H3
```

[VB.NET]

```
MNET6045SoftLtc(0, 0, &H3)
```

ライン0、デバイス0,軸パターン3h(軸0+軸1)にラッチ入力代行コマンドを発行します。

8－1－1－46. MNET6045SoftPcs

機能

PCS入力代行コマンド発行

書式

[Visual C++]

```
short __stdcall MNET6045SoftPcs(
                    short lineNo,
                    short devNo,
                    short axPtn);
```

[Visual Basic]

```
Declare Function MNET6045SoftPcs Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045SoftPcs Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン*** (8-2-1.軸パターン参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
MNET6045SoftPcs(0, 0, 0x3);
```

[Visual Basic]

```
MNET6045SoftPcs 0, 0, &H3
```

[VB.NET]

```
MNET6045SoftPcs(0, 0, &H3)
```

ライン0、デバイス0,軸パターン3h(軸0+軸1)にPCS入力代行コマンドを発行します。

8－1－1－47. MNET6045OutCsta

機能

CSTA(同時スタート)信号出力

- CSTA信号待ちとなっている軸はこの信号によりスタートします。
- CSTA信号待ちとするにはMNET6045SetPar関数で同期スタートモードを「同時スタート（CSTA）信号待ち」に設定した後、起動関数を実行してください。

書式

[Visual C++]

```
short __stdcall MNET6045OutCsta(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    short mode);
```

[Visual Basic]

```
Declare Function MNET6045OutCsta Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByVal mode As Integer)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045OutCsta Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByVal mode As Short)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axPtn	***軸パターン***（8-2-1.軸パターン参照）
mode	***モード*** 0:LSI内部のみ 1:外部出力含む

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

```
MNET6045OutCsta(0, 0, 0x3, 1);
```

[Visual Basic]

```
MNET6045OutCsta 0, 0, &H3, 1
```

[VB.NET]

```
MNET6045OutCsta(0, 0, &H3, 1)
```

ライン0、デバイス0,軸パターン3h(軸0+軸1)のCSTA(同時スタート)信号を出力します。またCSTA信号の外部出力も行なわれます。

8－1－1－48. MNET6045OutCstp

機能

CSTP(同時停止)信号出力

- CSTP信号が有効となっている軸はこの信号により停止します。
- CSTP信号有効/無効はMNET6045SetPar関数で設定します。

書式

[Visual C++]

```
short __stdcall MNET6045OutCstp(
                    short lineNo,
                    short devNo);
```

[Visual Basic]

```
Declare Function MNET6045OutCstp Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045OutCstp Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
MNET6045OutCstp(0, 0);
```

[Visual Basic]

```
MNET6045OutCstp 0, 0
```

[VB.NET]

```
MNET6045OutCstp(0, 0)
```

ライン0、デバイス0のCSTP(同時停止)信号を出力します。

8－1－1－49. MNET6045GetTermAbs

機能

最終到達位置データ取得

- プリレジスタ動作を含む動作が完了する位置を取得します。
- Jogや原点復帰などの最終位置が確定しない動作の完了位置は取得できません。

書式

[Visual C++]

```
short __stdcall MNET6045GetTermAbs(
                    short lineNo,
                    short devNo,
                    short axNo,
                    long* val);
```

[Visual Basic]

```
Declare Function MNET6045GetTermAbs Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByRef val As Long)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045GetTermAbs Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByRef val As Integer)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
val	***取得データ格納***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
long val;

MNET6045GetTermAbs(0, 0, 1, &val);
```

[Visual Basic]

```
Dim val As Long

MNET6045GetTermAbs 0, 0, 1, val
```

[VB.NET]

```
Dim val As Integer

MNET6045GetTermAbs(0, 0, 1, val)
```

ライン0、デバイス0,軸1の最終到達位置データを取得します。

8－1－1－50. MNET60450vridePos

機能

位置オーバーライド

- Ptp移動の場合、目標位置を変更します。
- PtpInc移動の場合、移動量を変更します。
- Ptp,PtpInc以外の動作ではエラーが返され、オーバーライドは行なわれません。

書式

[Visual C++]

```
short __stdcall MNET60450vridePos(
                    short lineNo,
                    short devNo,
                    short axNo,
                    long newPos);
```

[Visual Basic]

```
Declare Function MNET60450vridePos Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByVal newPos As Long)
                          As Integer
```

[VB.NET]

```
Declare Function MNET60450vridePos Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByVal newPos As Integer)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
newPos	***設定値***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
MNET60450vridePos(0, 0, 0, 10000);
```

[Visual Basic]

```
MNET60450vridePos 0, 0, 0, 10000
```

[VB.NET]

```
MNET60450vridePos(0, 0, 0, 10000)
```

ライン0、デバイス0、軸0の目標位置(移動量)を「10000」にオーバーライドします。

8－1－1－51. MNET6045ReadParFile

機能

パラメータファイル読み出し

・ 指定されたパラメータファイルを読み出し、読み出した値をレジスタに設定します。

書式

[Visual C++]

```
short __stdcall MNET6045ReadParFile(
                    short lineNo,
                    short devNo,
                    const char* filename);
```

[Visual Basic]

```
Declare Function MNET6045ReadParFile Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal filename As String)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045ReadParFile Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal filename As String)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
filename	***ファイル名***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

```
MNET6045ReadParFile(0, 0, “ABC.par”);
```

[Visual Basic]

```
MNET6045ReadParFile 0, 0, “ABC.par”
```

[VB.NET]

```
MNET6045ReadParFile(0, 0, “ABC.par)
```

パラメータファイル”ABC.par”を読み出しライン0、デバイス0に設定します。

8－1－1－52. MNET6045WriteParFile

機能

パラメータファイル書き込み

・ 指定されたファイルに環境設定などのレジスタデータを書き込みます。

書式

[Visual C++]

```
short __stdcall MNET6045WriteParFile(
                    short lineNo,
                    short devNo,
                    const char* filename);
```

[Visual Basic]

```
Declare Function MNET6045WriteParFile Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal filename As String)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045WriteParFile Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal filename As String)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
filename	***ファイル名***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

```
MNET6045WriteParFile(0, 0, “XYZ.txt”);
```

[Visual Basic]

```
MNET6045WriteParFile 0, 0, “XYZ.txt”
```

[VB.NET]

```
MNET6045WriteParFile(0, 0, “XYZ.txt”)
```

環境設定などのレジスタデータをファイル”XYZ.txt”に書き込みます。

8－1－1－53. MNET6045PulserPtpInc

機能

パルサー同期Ptp移動(Inc指定)

- 相対移動量指定のパルサー同期Ptp移動をスタートします。
- 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。
- **パルサー同期動作はFL定速動作のみで御利用いただけます。**

書式

[Visual C++]

```
short __stdcall MNET6045PulserPtpInc(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045PulserPtpInc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045PulserPtpInc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axPtn	***軸パターン***（8-2-1.軸パターン参照）
dst	***移動データ構造体***（8-2-5.移動データ構造体参照）

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};                 // 移動データ構造体定義,初期化

dst.Pos[0] = 10000;                     // 軸0移動量
dst.Pos[1] = -3000;                     // 軸1移動量
dst.StartMode = MNET_STAMODE_FL;        // スタートモード:FL定速

MNET6045PulserPtpInc(0, 0, 0x3, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst                 '/ 移動データ構造体定義,初期化

dst.Pos(0) = 10000                      '/ 軸0移動量
dst.Pos(1) = -3000                      '/ 軸1移動量
dst.StartMode = MNET_STAMODE_FL         '/ スタートモード:FL定速

MNET6045PulserPtpInc 0, 0, &H3, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst             '/ 移動データ構造体定義,初期化
dst.Initialize()                        '/ 構造体初期化

dst.Pos(0) = 10000                      '/ 軸0移動量
dst.Pos(1) = -3000                      '/ 軸1移動量
dst.StartMode = MNET_STAMODE_FL         '/ スタートモード:FL定速

MNET6045PulserPtpInc(0, 0, &H3, dst)
```

ライン0、デバイス0、軸パターン3h(軸0+軸1)のパルサー同期Ptp(Inc)移動を行います。

8－1－1－54. MNET6045PulserPtp

機能

パルサー同期Ptp移動(Abs指定)

- 絶対移動量指定のパルサー同期Ptp移動をスタートします。
- 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。
- **パルサー同期動作はFL定速動作のみで御利用いただけます。**

書式

[Visual C++]

```
short __stdcall MNET6045PulserPtp(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045PulserPtp Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045PulserPtp Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axPtn	***軸パターン***（8-2-1.軸パターン参照）
dst	***移動データ構造体***（8-2-5.移動データ構造体参照）

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};                 // 移動データ構造体定義,初期化

dst.Pos[0] = 20000;                     // 軸0目標位置
dst.Pos[2] = -6000;                     // 軸2目標位置
dst.StartMode = MNET_STAMODE_FL;        // スタートモード: FL定速

MNET6045PulserPtp(0, 0, 0x5, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst                 '/ 移動データ構造体定義,初期化

dst.Pos(0) = 20000                      '/ 軸0目標位置
dst.Pos(2) = -6000                      '/ 軸2目標位置
dst.StartMode = MNET_STAMODE_FL         '/ スタートモード: FL定速

MNET6045PulserPtp 0, 0, &H5, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst             '/ 移動データ構造体定義,初期化
dst.Initialize()                        '/ 構造体初期化

dst.Pos(0) = 20000                      '/ 軸0目標位置
dst.Pos(2) = -6000                      '/ 軸2目標位置
dst.StartMode = MNET_STAMODE_FL         '/ スタートモード: FL定速

MNET6045PulserPtp(0, 0, &H5, dst)
```

ライン0、デバイス0、軸パターン5h(軸0+軸2)のパルサー同期Ptp(Abs)移動を行います。

8－1－1－55. MNET6045PulserLineInc

機能

パルサー同期直線補間移動(Inc指定)

- 速度はベクトル速度で指定します。
- 速度データは軸パターンに関わらず速度データ構造体の0軸目に設定します。
- **パルサー同期動作はFL定速動作のみで御利用いただけます。**

書式

[Visual C++]

```
short __stdcall MNET6045PulserLineInc(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst,
                    TMNET6045Spd* spd);
```

[Visual Basic]

```
Declare Function MNET6045PulserLineInc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst,
                    ByRef spd As TMNET6045Spd)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045PulserLineInc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst,
                    ByRef spd As TMNET6045Spd)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン*** (8-2-1.軸パターン参照)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)
spd	***速度データ構造体*** (8-2-4.速度データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};              // 移動データ構造体定義,初期化
TMNET6045Spd spd = {0};              // 速度データ構造体定義,初期化

dst.Pos[0] = 10000;                  // 軸0移動量
dst.Pos[2] = -3000;                  // 軸2移動量
dst.StartMode = MNET_STAMODE_FL;     // スタートモード: FL定速

spd.SpdL[0] = 2000;                  // FL速度設定

MNET6045PulserLineInc(0, 0, 0x5, &dst, &spd);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst              '/ 移動データ構造体定義,初期化
Dim spd As TMNET6045Spd              '/ 速度データ構造体定義,初期化

dst.Pos(0) = 10000                   '/ 軸0移動量
dst.Pos(2) = -3000                   '/ 軸2移動量
dst.StartMode = MNET_STAMODE_FL      '/ スタートモード: FL定速

spd.SpdL(0) = 2000                   '/ FL速度設定

MNET6045PulserLineInc 0, 0, &H5, dst, spd
```

[VB.NET]

```
  Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
  Dim spd As New TMNET6045Spd          '/ 速度データ構造体定義,初期化
  dst.Initialize()                     '/ 構造体初期化
  spd.Initialize()                     '/ 構造体初期化

  dst.Pos(0) = 10000                   '/ 軸0移動量
  dst.Pos(2) = -3000                   '/ 軸2移動量
  dst.StartMode = MNET_STAMODE_FL      '/ スタートモード: FL定速
  spd.SpdL(0) = 2000                   ‘/ FL速度設定

  MNET6045PulserLineInc(0, 0, &H5, dst, spd)
```

ライン0、デバイス0、軸パターン5h(軸0+軸2)のパルサー同期Inc指定直線補間移動を行います。

8－1－1－56. MNET6045PulserLine

機能

パルサー同期直線補間移動(Abs指定)

- 速度はベクトル速度で指定します。
- 速度データは軸パターンに関わらず速度データ構造体の0軸目に設定します。
- **パルサー同期動作はFL定速動作のみで御利用いただけます。**

書式

[Visual C++]

```
short __stdcall MNET6045PulserLine(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst,
                    TMNET6045Spd* spd);
```

[Visual Basic]

```
Declare Function MNET6045PulserLine Lib "PPCI_L112.dll"(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst,
                    ByRef spd As TMNET6045Spd)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045PulserLine Lib "PPCI_L112.dll"(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst,
                    ByRef spd As TMNET6045Spd)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axPtn	***軸パターン***（8-2-1.軸パターン参照）
dst	***移動データ構造体***（8-2-5.移動データ構造体参照）
spd	***速度データ構造体***（8-2-4.速度データ構造体参照）

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};              // 移動データ構造体定義,初期化
TMNET6045Spd spd = {0};              // 速度データ構造体定義,初期化

dst.Pos[1] = 10000;                  // 軸1目標位置
dst.Pos[2] = -3000;                  // 軸2目標位置
dst.StartMode = MNET_STAMODE_FL;     // スタートモード: FL定速

spd.SpdL[0] = 2000;                  // FL速度設定

MNET6045PulserLine(0, 0, 0x6, &dst, &spd);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst              '/ 移動データ構造体定義,初期化
Dim spd As TMNET6045Spd              '/ 速度データ構造体定義,初期化

dst.Pos(1) = 10000                   '/ 軸1目標位置
dst.Pos(2) = -3000                   '/ 軸2目標位置
dst.StartMode = MNET_STAMODE_FL      '/ スタートモード: FL定速

spd.SpdL(0) = 2000                   '/ FL速度設定

MNET6045PulserLine 0, 0, &H6, dst, spd
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
Dim spd As New TMNET6045Spd          '/ 速度データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化
spd.Initialize()                     '/ 構造体初期化

dst.Pos(1) = 10000                   '/ 軸1目標位置
dst.Pos(2) = -3000                   '/ 軸2目標位置
dst.StartMode = MNET_STAMODE_FL      '/ スタートモード: FL定速

spd.SpdL(0) = 2000                   '/ FL速度設定

MNET6045PulserLine(0, 0, &H6, dst, spd)
```

ライン0、デバイス0、軸パターン6h(軸1+軸2)のパルサー同期Abs指定直線補間移動を行います。

8－1－1－57. MNET6045PulserArcInc

機能

パルサー同期円弧補間移動(Inc指定)

- 速度データは軸パターンに関わらず速度データ構造体の0軸目に設定します。
- 回転方向データは軸パターンに関わらず移動データ構造体の0軸目に設定します。
- **パルサー同期動作はFL定速動作のみで御利用いただけます。**

書式

[Visual C++]

```
short __stdcall MNET6045PulserArcInc(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst,
                    TMNET6045Spd* spd);
```

[Visual Basic]

```
Declare Function MNET6045PulserArcInc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst,
                    ByRef spd As TMNET6045Spd)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045PulserArcInc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst,
                    ByRef spd As TMNET6045Spd)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン*** (8-2-1.軸パターン参照)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)
spd	***速度データ構造体*** (8-2-4.速度データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};              // 移動データ構造体
TMNET6045Spd spd = {0};              // 速度データ構造体

dst.Pos[0] = 1000;                   // 軸0目標相対位置
dst.Pos[2] = 300;                    // 軸2目標相対位置
dst.Center[0] = 600;                 // 軸0中心相対位置
dst.Center[2] = 0;                   // 軸2中心相対位置
dst.Direction[0] = MNET_DIR_CW;      // 回転方向:CW
dst.StartMode = MNET_STAMODE_FL;     // スタートモード: FL定速

spd.SpdL[0] = 2000;                  // FL速度設定

MNET6045PulserArcInc(0, 0, 0x5, &dst, &spd);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst              '/ 移動データ構造体定義,初期化
Dim spd As TMNET6045Spd              '/ 速度データ構造体定義,初期化

dst.Pos(0) = 1000                    '/ 軸0目標相対位置
dst.Pos(2) = 300                     '/ 軸2目標相対位置
dst.Center(0) = 600                  '/ 軸0中心相対位置
dst.Center(2) = 0                    '/ 軸2中心相対位置
dst.Direction(0) = MNET_DIR_CW       '/ 回転方向:CW
dst.StartMode = MNET_STAMODE_FL      '/ スタートモード: FL定速

spd.SpdL(0) = 2000                   '/ FL速度設定

MNET6045PulserArcInc 0, 0, &H5, dst, spd
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
Dim spd As New TMNET6045Spd          '/ 速度データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化
spd.Initialize()                     '/ 構造体初期化

dst.Pos(0) = 1000                    '/ 軸0目標相対位置
dst.Pos(2) = 300                     '/ 軸2目標相対位置
dst.Center(0) = 600                  '/ 軸0中心相対位置
dst.Center(2) = 0                    '/ 軸2中心相対位置
dst.Direction(0) = MNET_DIR_CW       '/ 回転方向:CW
dst.StartMode = MNET_STAMODE_FL      '/ スタートモード: FL定速

spd.SpdL(0) = 2000                   '/ FL速度設定

MNET6045PulserArcInc(0, 0, &H5, dst, spd)
```

ライン0、デバイス0、軸パターン5h(軸0+軸2)のパルサー同期Inc指定円弧補間移動を行います。

Motionnet®
Remote I/O & Remote Motion

8－1－1－58. MNET6045PulserArc

機能

パルサー同期円弧補間移動(Abs指定) (軸パターン指定)

- 速度データは軸パターンに関わらず速度データ構造体の0軸目に設定します。
- 回転方向データは軸パターンに関わらず移動データ構造体の0軸目に設定します。
- **パルサー同期動作はFL定速動作のみで御利用いただけます。**

書式

[Visual C++]

```
short __stdcall MNET6045PulserArc(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst,
                    TMNET6045Spd* spd);
```

[Visual Basic]

```
Declare Function MNET6045PulserArc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst,
                    ByRef spd As TMNET6045Spd)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045PulserArc Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst,
                    ByRef spd As TMNET6045Spd)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0 始まり)
axPtn	***軸パターン*** (8-2-1.軸パターン参照)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)
spd	***速度データ構造体*** (8-2-4.速度データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESS が返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};            // 移動データ構造体定義,初期化
TMNET6045Spd spd = {0};            // 速度データ構造体定義,初期化

dst.Pos[1] = 1000;                  // 軸1目標位置
dst.Pos[2] = 300;                   // 軸2目標位置
dst.Center[1] = 600;                // 軸1中心位置
dst.Center[2] = 0;                  // 軸2中心位置
dst.Direction[0] = MNET_DIR_CW;     // 回転方向:CW
dst.StartMode = MNET_STAMODE_FL;    // スタートモード: FL定速

spd.SpdL[0] = 2000;                 // FL速度設定

MNET6045PulserArc(0, 0, 0x6, &dst, &spd);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst             '/ 移動データ構造体定義,初期化
Dim spd As TMNET6045Spd             '/ 速度データ構造体定義,初期化

dst.Pos(1) = 1000                   '/ 軸1目標位置
dst.Pos(2) = 300                    '/ 軸2目標位置
dst.Center(1) = 600                 '/ 軸1中心位置
dst.Center(2) = 0                   '/ 軸2中心位置
dst.Direction(0) = MNET_DIR_CW      '/ 回転方向:CW
dst.StartMode = MNET_STAMODE_FL     '/ スタートモード: FL定速

spd.SpdL(0) = 2000                  '/ FL速度設定

MNET6045PulserArc 0, 0, &H6, dst, spd
```

[VB.NET]

```
  Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
  Dim spd As New TMNET6045Spd          '/ 速度データ構造体定義,初期化
  dst.Initialize()                     '/ 構造体初期化
  spd.Initialize()                     '/ 構造体初期化

  dst.Pos(1) = 1000                    '/ 軸1目標位置
  dst.Pos(2) = 300                     '/ 軸2目標位置
  dst.Center(1) = 600                  '/ 軸1中心位置
  dst.Center(2) = 0                    '/ 軸2中心位置
  dst.Direction(0) = MNET_DIR_CW       '/ 回転方向:CW
  dst.StartMode = MNET_STAMODE_FL      '/ スタートモード: FL定速

  spd.SpdL(0) = 2000                   '/ FL速度設定

  MNET6045PulserArc(0, 0, &H6, dst, spd)
```

ライン0、デバイス0、軸パターン6h(軸1+軸2)のパルサー同期Abs指定円弧補間移動を行います。

8－1－1－59. MNET6045PulserContinue

機能

パルサー同期連続移動

連続移動とは Jog 動作のような移動量指定の無い、指定方向へ動き続ける動作です。

動作速度は MNET6045SetSpeed 関数、MNET6045SetSpeedNo 関数で設定した速度となります。

パルサー同期動作は FL 定速動作のみで御利用いただけます。

書式

[Visual C++]

```
short __stdcall MNET6045PulserContinue(
                    short lineNo,
                    short devNo,
                    short axPtn,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045PulserContinue Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axPtn As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045PulserContinue Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axPtn As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axPtn	***軸パターン*** (8-2-1.軸パターン参照)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};              // 移動データ構造体定義,初期化

dst.Direction[0] = MNET_DIR_PLUS;    // 軸0動作方向
dst.Direction[2] = MNET_DIR_MINUS;   // 軸2動作方向
dst.StartMode = MNET_STAMODE_FL;     // スタートモード: FL定速

MNET6045PulserContinue(0, 0, 0x5, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst              '/ 移動データ構造体定義,初期化

dst.Direction(0) = MNET_DIR_PLUS     '/ 軸0動作方向
dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_FL      '/ スタートモード: FL定速

MNET6045PulserContinue 0, 0, &H5, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化

dst.Direction(0) = MNET_DIR_PLUS     '/ 軸0動作方向
dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_FL      '/ スタートモード: FL定速

MNET6045PulserContinue(0, 0, &H5, dst)
```

ライン0、デバイス0、軸パターン5h(軸0+軸2)のパルサー同期連続移動を行います。

8－1－1－60. MNET6045PulserPtpIncNo

機能

パルサー同期Ptp移動(Inc指定) (軸番号指定)

- 相対移動量指定のパルサー同期Ptp移動をスタートします。
- 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。
- **パルサー同期動作はFL定速動作のみで御利用いただけます。**

書式

[Visual C++]

```
short __stdcall MNET6045PulserPtpIncNo (
                    short lineNo,
                    short devNo,
                    short axNo,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045PulserPtpIncNo Lib "PPCI_L112.dll"(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045PulserPtpIncNo Lib "PPCI_L112.dll"(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};                 // 移動データ構造体定義,初期化

dst.Pos[0] = 10000;                     // 軸0移動量
dst.StartMode = MNET_STAMODE_FL;        // スタートモード:FL定速

MNET6045PulserPtpIncNo(0, 0, 0, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst                 '/ 移動データ構造体定義,初期化

dst.Pos(0) = 10000                      '/ 軸0移動量
dst.StartMode = MNET_STAMODE_FL         '/ スタートモード:FL定速

MNET6045PulserPtpIncNo 0, 0, 0, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst             '/ 移動データ構造体定義,初期化
dst.Initialize()                        '/ 構造体初期化

dst.Pos(0) = 10000                      '/ 軸0移動量
dst.StartMode = MNET_STAMODE_FL         '/ スタートモード:FL定速

MNET6045PulserPtpIncNo(0, 0, 0, dst)
```

ライン0、デバイス0、軸0のパルサー同期Ptp(Inc)移動を行います。

8－1－1－61. MNET6045PulserPtpNo

機能

パルサー同期Ptp移動(Abs指定) (軸番号指定)

- 絶対移動量指定のパルサー同期Ptp移動をスタートします。
- 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。
- **パルサー同期動作はFL定速動作のみで御利用いただけます。**

書式

[Visual C++]

```
short __stdcall MNET6045PulserPtpNo (
                    short lineNo,
                    short devNo,
                    short axNo,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045PulserPtpNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045PulserPtpNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};                 // 移動データ構造体定義,初期化

dst.Pos[2] = -6000;                     // 軸2目標位置
dst.StartMode = MNET_STAMODE_FL;        // スタートモード: FL定速

MNET6045PulserPtpNo(0, 0, 2, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst                 '/ 移動データ構造体定義,初期化

dst.Pos(2) = -6000                      '/ 軸2目標位置
dst.StartMode = MNET_STAMODE_FL         '/ スタートモード: FL定速

MNET6045PulserPtpNo 0, 0, 2, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst             '/ 移動データ構造体定義,初期化
dst.Initialize()                        '/ 構造体初期化

dst.Pos(2) = -6000                      '/ 軸2目標位置
dst.StartMode = MNET_STAMODE_FL         '/ スタートモード: FL定速

MNET6045PulserPtpNo(0, 0, 2, dst)
```

ライン0、デバイス0、軸2のパルサー同期Ptp(Abs)移動を行います。

Motionnet®
Remote I/O & Remote Motion

8－1－1－62. MNET6045PulserContinueNo

機能

パルサー同期連続移動 （軸番号指定）

- 連続移動とはJog動作のような移動量指定の無い、指定方向へ動き続ける動作です。
- 動作速度はMNET6045SetSpeed関数、MNET6045SetSpeedNo関数で設定した速度となります。
- **パルサー同期動作はFL定速動作のみで御利用いただけます。**

書式

[Visual C++]

```
short __stdcall MNET6045PulserContinueNo (
                    short lineNo,
                    short devNo,
                    short axNo,
                    TMNET6045Dst* dst);
```

[Visual Basic]

```
Declare Function MNET6045PulserContinueNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByRef dst As TMNET6045Dst)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045PulserContinueNo Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByRef dst As TMNET6045Dst)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号*** (0始まり)
dst	***移動データ構造体*** (8-2-5.移動データ構造体参照)

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
TMNET6045Dst dst = {0};              // 移動データ構造体定義,初期化

dst.Direction[2] = MNET_DIR_MINUS;   // 軸2動作方向
dst.StartMode = MNET_STAMODE_FL;     // スタートモード: FL定速

MNET6045PulserContinueNo(0, 0, 2, &dst);
```

[Visual Basic]

```
Dim dst As TMNET6045Dst              '/ 移動データ構造体定義,初期化

dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_FL      '/ スタートモード: FL定速

MNET6045PulserContinueNo 0, 0, 2, dst
```

[VB.NET]

```
Dim dst As New TMNET6045Dst          '/ 移動データ構造体定義,初期化
dst.Initialize()                     '/ 構造体初期化

dst.Direction(2) = MNET_DIR_MINUS    '/ 軸2動作方向
dst.StartMode = MNET_STAMODE_FL      '/ スタートモード: FL定速

MNET6045PulserContinueNo(0, 0, 2, dst)
```

ライン0、デバイス0、軸2のパルサー同期連続移動を行います。

8－1－1－63. MNET6045GetINTFactor

機能

割込要因取得

- 指定したボード全4軸の割込要因データ(エラー割込とイベント割込)を取得します。
- 本関数実行により割込信号はOFFします。
- 本関数実行により割込要因データ(エラー割込とイベント割込)はリセットされます。
- エラー割込要因については「8-3-2-1. エラー割込要因」を参照ください。
- イベント割込要因については「8-3-2-2. イベント割込要因」を参照ください。

書式

[Visual C++]

```
short __stdcall MNET6045GetINTFactor(
                    short lineNo,
                    short devNo,
                    long* rest,
                    long* rist);
```

[Visual Basic]

```
Declare Function MNET6045GetINTFactor Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByRef rest As Long,
                    ByRef rist As Long)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045GetINTFactor Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByRef rest As Integer,
                    ByRef rist As Integer)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
rest	***エラー割込要因格納配列アドレス***
rist	***イベント割込要因格納配列アドレス***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

```
long rest[4] = {0};                    // エラー割込要因格納配列
long rist[4] = {0};                    // イベント割込要因格納配列

MNET6045GetINTFactor(0, 0, rest, rist);
```

[Visual Basic]

```
Dim rest(0 to 3) As Long               '/ エラー割込要因格納配列
Dim rist(0 to 3) As Long               '/ イベント割込要因格納配列

MNET6045GetINTFactor 0, 0, rest(0), rist(0)
```

[VB.NET]

```
Dim rest(3) As Integer                 '/ エラー割込要因格納配列
Dim rist(3) As Integer                 '/ イベント割込要因格納配列

MNET6045GetINTFactor(0, 0, rest(0), rist(0))
```

ライン0、デバイス0の割込要因データを取得します。

8－1－2. 基本関数詳細

8－1－2－1. MNET6045WReg

機能

レジスタ書込み

書式

[Visual C++]

```
short __stdcall MNET6045WReg(
                    short lineNo,
                    short devNo,
                    short axNo,
                    unsigned short cmd,
                    long val);
```

[Visual Basic]

```
Declare Function MNET6045WReg Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByVal cmd As Integer,
                    ByVal val As Long)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045WReg Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByVal cmd As Short,
                    ByVal val As Integer)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axNo	***軸番号***(0始まり)
cmd	***レジスタ書込みコマンド***
val	***設定値***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

```
MNET6045WReg(0, 0, 1, WRCUN1, 1000);
```

[Visual Basic]

```
MNET6045WReg 0, 0, 1, WRCUN1, 1000
```

[VB.NET]

```
MNET6045Wreg(0, 0, 1, WRCUN1, 1000)
```

ライン0、デバイス0,軸1のRCUN1レジスタに1000を設定します。

8－1－2－2. MNET6045RReg

機能

レジスタ読出し

書式

[Visual C++]

```
short __stdcall MNET6045RReg(
                    short lineNo,
                    short devNo,
                    short axNo,
                    unsigned short cmd,
                    long* val);
```

[Visual Basic]

```
Declare Function MNET6045RReg Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByVal cmd As Integer,
                    ByRef val As Long)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045RReg Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByVal cmd As Short,
                    ByRef val As Integer)
                          As Short
```

パラメータ

lineNo	***ライン番号***（0始まり）
devNo	***デバイス番号***（0始まり）
axNo	***軸番号***(0始まり)
cmd	***レジスタ読出しコマンド***
val	***取得値格納***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。（8-3-3-12. 戻り値参照）

使用例

[Visual C++]

```
long val;

MNET6045RReg(0, 0, 1, RRCUN1, &val);
```

[Visual Basic]

```
Dim val As Long

MNET6045RReg 0, 0, 1, RRCUN1, val
```

[VB.NET]

```
Dim val As Integer

MNET6045RReg(0, 0, 1, RRCUN1, val)
```

ライン0、デバイス0,軸1のRCUN1レジスタの値を取得します。

8－1－2－3. MNET6045WCmd

機能

コマンド書き込み

書式

[Visual C++]

```
short __stdcall MNET6045WCmd(
                    short lineNo,
                    short devNo,
                    short axNo,
                    unsigned short cmd);
```

[Visual Basic]

```
Declare Function MNET6045WCmd Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByVal cmd As Integer)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045WCmd Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByVal cmd As Short)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号***(0始まり)
cmd	**コマンド**

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
MNET6045WCmd(0, 0, 1, STOP);
```

[Visual Basic]

```
MNET6045WCmd 0, 0, 1, STOP
```

[VB.NET]

```
MNET6045WCmd(0, 0, 1, STOP)
```

ライン0、デバイス0,軸1にSTOPコマンドを書き込みます。

8－1－2－4. MNET6045WPort

機能

ポート書き込み

書式

[Visual C++]

```
short __stdcall MNET6045WPort(
                    short lineNo,
                    short devNo,
                    short axNo,
                    unsigned short val);
```

[Visual Basic]

```
Declare Function MNET6045WPort Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByVal val As Integer)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045WPort Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByVal val As Short)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号***(0始まり)
val	***設定値***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
MNET6045WPort(0, 0, 1, 0x3);
```

[Visual Basic]

```
MNET6045WPort 0, 0, 1, &H3
```

[VB.NET]

```
MNET6045Wport(0, 0, 1, &H3)
```

ライン0、デバイス0,軸1のポートに3hを書き込みます。

Motionnet® Remote I/O & Remote Motion

8－1－2－5. MNET6045RSts

機能

ステータス読出し

書式

[Visual C++]

```
short __stdcall MNET6045RSts(
                    short lineNo,
                    short devNo,
                    short axNo,
                    long* val);
```

[Visual Basic]

```
Declare Function MNET6045RSts Lib “PPCI_L112.dll”(
                    ByVal lineNo As Integer,
                    ByVal devNo As Integer,
                    ByVal axNo As Integer,
                    ByRef val As Long)
                          As Integer
```

[VB.NET]

```
Declare Function MNET6045RSts Lib “PPCI_L112.dll”(
                    ByVal lineNo As Short,
                    ByVal devNo As Short,
                    ByVal axNo As Short,
                    ByRef val As Integer)
                          As Short
```

パラメータ

lineNo	***ライン番号*** (0始まり)
devNo	***デバイス番号*** (0始まり)
axNo	***軸番号***(0始まり)
val	***取得値格納***

戻り値

正常に終了した場合は、MNET_SUCCESSが返されます。正常な処理が行われなかった場合はその原因がエラーコードとして返されます。 (8-3-3-12. 戻り値参照)

使用例

[Visual C++]

```
long val;

MNET6045RSts(0, 0, 1, &val);
```

[Visual Basic]

```
Dim val As Long

MNET6045RSts 0, 0, 1, val
```

[VB.NET]

```
Dim val As Integer

MNET6045RSts(0, 0, 1, val)
```

ライン0、デバイス0,軸1のステータスデータを取得します。

8－2. 機能操作説明

8－2－1. 軸パターン

本DLLでは「軸パターン」データを多く利用します。
軸パターンとは4軸を4ビットのデータで表し操作対象軸の選択を行なうデータです。
軸パターンでの指令により、任意の複数軸に対し同時に指令を出すことが可能となっています。

[例]

軸選択	軸パターン値
軸3 軸2 軸1 軸0 0 1 0 1	5h
軸3 軸2 軸1 軸0 1 1 1 1	Fh
軸3 軸2 軸1 軸0 1 0 0 0	8h
軸3 軸2 軸1 軸0 0 0 0 1	1h

Motionnet®
Remote I/O & Remote Motion

8−2−2. データの取得

8−2−2−1. 状態取得

入出力信号状態、ステータスビット、および軸動作状態などについてはMNET6045GetBit関数、またはMNET6045GetBitStruct関数を使用し取得が可能です。

[MNET6045GetBit 関数]

対象ビット(データ)を指定し、その項目のデータを取得します。

記述例

[Visual C++]

```
short data;

MNET6045GetBit(0, 0, 1, MNET6045_BIT_SCP1, &data);
```

[Visual Basic]

```
Dim data As Integer

MNET6045GetBit 0, 0, 1, MNET6045_BIT_SCP1, data
```

[VB.NET]

```
Dim data As Short

MNET6045GetBit(0, 0, 1, MNET6045_BIT_SCP1, data)
```

ライン0,デバイス0,軸1のビットSCP1の状態(コンパレータ1成立状態)を変数dataに取得します。

Motionnet®
Remote I/O & Remote Motion

[MNET6045GetBitStruct 関数]

ビットデータ構造体にビット(データ)を一括で取得します。

使用例

[Visual C++]

```
TMNET6045BitData bit[4] = {0};       // ビットデータ構造体定義

MNET6045GetBitStruct(0, 0, 1, &bit[1]);  // ライン0,デバイス0,軸1のビットデータを一括取得
if (bit[1].Spel != 0)                // Spel (プラスEL)ビットによる判定
  m_pel.SetCheck(TRUE);              // ON時:チェックON
else
  m_pel.SetCheck(FALSE);             // OFF時: チェックOFF

if (bit[1].Salm  != 0)               // Salm(ALM)ビットによる判定
  m_alm.SetCheck(TRUE);              // ON時: チェックON
else
  m_alm.SetCheck(FALSE);             // OFF時: チェックOFF
```

[Visual Basic]

```
Dim bit(0 to 3) As TMNET6045BitData   '/ ビットデータ構造体定義

MNET6045GetBitStruct 0, 0, 1, bit(1) '/ ライン0,デバイス0,軸1のビットデータを一括取得
if (bit(1).Spel <> 0) then           '/ Spel (プラスEL)ビットによる判定
  shapePel.FillColor = &HFF&         '/ ON時:明赤色
else
  shapePel.FillColor = &H80&         '/ OFF時:暗赤色
endif

if (bit(1).Salm  <> 0) then          ' Salm(ALM)ビットによる判定
  shapeAlm.FillColor = &HFF&         '/ ON時:明赤色
else
  shapeAlm.FillColor = &H80&         '/ OFF時:暗赤色
endif
```

[VB.NET]

```
Dim bit(0 to 3) As TMNET6045BitData   '/ ビットデータ構造体定義

MNET6045GetBitStruct 0, 0, 1, bit(1) '/ ライン0,デバイス0,軸1のビットデータを一括取得
if (bit(1).Spel <> 0) then           '/ Spel (プラスEL)ビットによる判定
  shapePel.BackColor = System.Drawing.Color.Red         '/ ON時:明赤色
else
  shapePel.BackColor = System.Drawing.Color.DarkRed     '/ OFF時:暗赤色
endif

if (bit(1).Salm  <> 0) then          '/ Salm(ALM)ビットによる判定
  shapeAlm.BackColor = System.Drawing.Color.Red         '/ ON時:明赤色
else
  shapeAlm.BackColor = System.Drawing.Color.DarkRed     ' OFF時:暗赤色
endif
```

8−2−2−2. カウンタ取得

カウンタデータの取得は、MNET6045GetCounter関数で行います。
本ボードにカウンタは各軸×4本あり、それぞれのカウンタ入力源はMNET6045SetPar関数で設定可能です。

記述例

[Visual C++]

```
long data;

MNET6045GetCounter(0, 0, 1, MNET_SEL_COUNTER1, &data);
```

[Visual Basic]

```
Dim data As Long

MNET6045GetCounter 0, 0, 1, MNET_SEL_COUNTER1, data
```

[VB.NET]

```
Dim data As Integer

MNET6045GetCounter(0, 0, 1, MNET_SEL_COUNTER1, data)
```

ライン0,デバイス0,軸1のカウンタ1(指令位置カウンタ)のデータを変数dataに取得します。

8−2−2−3. 停止確認

軸の停止確認は主に以下の2つの方法が利用できます。

[MNET6045GetBitStruct関数、MNET6045GetBit関数]
指定した軸の状態に関連するビットやデータなどを取得し判定します。

[MNET6045StopChk関数]
関数内部で軸の状態に関連するビットやデータが取得され、軸パターンで指定した軸の停止/動作中判別、正常停止/異常停止などが判定された結果を取得できます。

8−2−2−4. エラー割込要因取得

エラー割込要因の取得はMNET6045GetINTFactor関数を使用します。
エラー割込要因データは読出しによりリセットされ0となります。
(本関数は割込が有効に設定されていない場合でも利用できます)

8−2−2−5. イベント割込要因取得

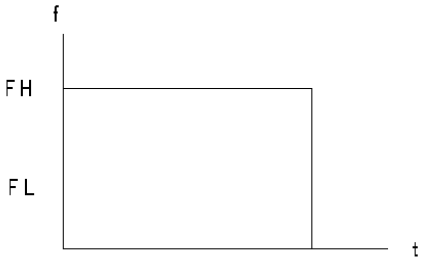
イベント割込要因の取得はMNET6045GetINTFactor関数を使用します。
イベント割込要因データは読出しによりリセットされ0となります。
またイベント割込要因の設定はMNET6045SetPar関数で行います。
(本関数は割込が有効に設定されていない場合でも利用できます)

8−2−3. スタートモード

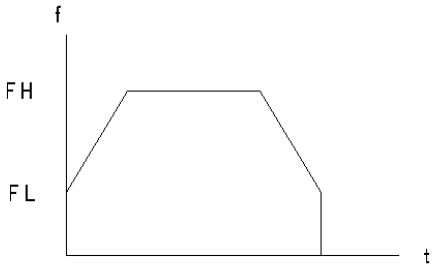
スタートモードは動作の速度パターンを表しており、以下の３通りのモードがあります。
速度指定時などに設定するデータとその識別子は次の通りとなります。

識別子	値	動作
MNET_STAMODE_FL	50h	ＦＬ定速動作
MNET_STAMODE_FH	51h	ＦＨ定速動作
MNET_STAMODE_UD	53h	加減速動作

ＦＬ定速動作
速度データ構造体メンバー「SpdL」に指定した速度での定速動作となります。



ＦＨ定速動作
速度データ構造体メンバー「SpdH」に指定した速度での定速動作となります。



加減速動作
速度データ構造体メンバー「SpdL」速度から加速し「SpdH」速度での定速動作の後、減速して停止します。
加速度はメンバー「Acc」、減速度は「Dec」で指定します。

Motionnet®
Remote I/O & Remote Motion

8−2−4. 速度データ構造体

速度データの指定に使用します。

書式

[Visual C++]

```
struct TMNET6045Spd {
  double SpdL[4];            // FL速度
  double SpdH[4];            // FH速度
  double Acc[4];             // 加速データ
  double Dec[4];             // 減速データ
  double SAccSect[4];        // 加速S字範囲
  double SDecSect[4];        // 減速S字範囲
  short AdcSpc[4];           // 加減速指定方法
  short SMode[4];            // S字モード
};
```

[Visual Basic]

```
Type TMNET6045Spd
  SpdL(0 to 3) As Double      '/ FL速度
  SpdH(0 to 3) As Double      '/ FH速度
  Acc(0 to 3) As Double       '/ 加速データ
  Dec(0 to 3) As Double       '/ 減速データ
  SAccSect(0 to 3) As Double  '/ 加速S字範囲
  SDecSect(0 to 3) As Double  '/ 減速S字範囲
  AdcSpc(0 to 3) As Integer   '/ 加減速指定方法
  SMode(0 to 3) As Integer    '/ S字モード
End Type
```

[VB.NET]

```
<StructLayout(LayoutKind.Sequential)> Structure TMNET6045Spd
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim SpdL() As Double '/ FL速度
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim SpdH() As Double '/ FH速度
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim Acc() As Double '/ 加速時間
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim Dec() As Double '/ 減速時間
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim SAccSect() As Double '/ 加速S字範囲
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim SDecSect() As Double '/ 減速S字範囲
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim AdcSpc() As Short '/ 加減速指定方法
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim SMode() As Short '/ S字モード
  Public Sub Initialize()
          ReDim SpdL(3)
          ReDim SpdH(3)
          ReDim Acc(3)
          ReDim Dec(3)
          ReDim SAccSect(3)
          ReDim SDecSect(3)
          ReDim AdcSpc(3)
          ReDim SMode(3)
  End Sub
End Structure
```


Motionnet® Remote I/O & Remote Motion

[メンバー]

メンバー	内容	設定の必要性	設定値
SpdL[4]	対象軸のFL速度	速度設定時(スタートモードがFL,FH,加減速の場合)	PPS
SpdH[4]	対象軸のFH速度	速度設定時(スタートモードがFH,加減速の場合)	PPS
Acc[4]	加速データ	加減速設定時	AdcSpcの指定に応じ、レートまたは時間として扱われます
Dec[4]	減速データ	加減速設定時	AdcSpcの指定に応じ、レートまたは時間として扱われます
SAccSect[4]	加速S字区間	加減速設定時(S字動作時)	設定値0の場合、直線部分のないS字動作となります
SDecSect[4]	減速S字区間	加減速設定時(S字動作時)	設定値0の場合、直線部分のないS字動作となります
AdcSpc[4]	加減速度指定方法	加減速設定時	0 : レート(1秒あたりの増減分パルス) 1 : 時間(秒)
SMode[4]	S字有無	加減速設定時	0 : S字無し 1 : S字有り

・メンバーに[4]の付くデータは配列メンバーです。データは各軸ごとに設定します。
・対象軸以外への設定は不要です。（対象軸以外の設定は無視されます(動作に影響しません)）
・直線補間時および円弧補間時のデータは対象軸に関わらず配列番号:0の要素に設定ください。

設定されるデータ

設定されるデータはスタートモード別に以下のようになります。
また後述の「設定上の注意」を必ずご確認ください。

FL 定速の場合

SpdL : FL速度 必ず設定が必要です。(速度0は速度設定エラーとなります)

上記以外のメンバーのデータは無視されます。(これ以外のデータは設定されません)

FH 定速の場合

SpdL : FL速度 (速度0の場合,設定可能な最小値が設定されます)
SpdH : FH速度 必ず設定が必要です。(速度0は速度設定エラーとなります)

上記以外のメンバーのデータは無視されます。(これ以外のデータは設定されません)

加減速の場合

SpdL : FL速度 必ず設定が必要です。(速度0の場合,設定可能な最小値が設定されます)
SpdH : FH速度 必ず設定が必要です。(速度0は速度設定エラーとなります。FL≦FHである必要があります)
Acc : 加速データ 必ず設定が必要です。
Dec : 減速データ 必ず設定が必要です。
SAccSect : 加速S字範囲
SDecSect : 減速S字範囲
AdcSpc : 加減速指定方法 必ず設定が必要です。
SMode : S字使用/未使用 必ず設定が必要です。

S字範囲の設定は必須ではありません。

設定上の注意

① Visual C++での使用の場合
構造体定義時にメンバーを下記のように初期化(0クリア)してください。

```
TMNET6045Spd spd = {0};   // 構造体定義と初期化(0クリア)
```

初期化を行わない場合、メンバーに残った不定データにより正しい動作が行われない可能性があります。

② VB.NET での使用の場合
構造体定義時にメンバーを下記のように初期化してください。

```
Dim spd As New TMNET6045Spd    ’速度データ構造体
spd.Initialize()               ’構造体初期化
```

③ DLL内部の速度計算について
SpdL/SpdHに設定された速度は2つの速度のうち大きい速度を基準に、速度倍率(速度設定できる速度間隔)が決まりLSIに設定される速度が演算されます。設定される速度は演算により丸められる可能性があることをご承知おきください。詳細については速度倍率についての項を参照ください。

④ 動作起動関数の引数「スタートモード」
直近に実行されたMNET6045SetSpeed関数のスタートモードにより、動作起動関数(MNET6045Ptp他(補間関数を除く))実行時に指定できるスタートモードには制限があり、以下のようになっております。
このデータはDLL内で各軸毎に記憶/管理されております。

MNET6045SetSpeed 関数での スタートモード指定	動作起動関数で指定可能な スタートモード
MNET_STAMODE_FL(FL 定速)	MNET_STAMODE_FL(FL 定速)
MNET_STAMODE_FH(FH 定速)	MNET_STAMODE_FL(FL 定速) MNET_STAMODE_FH(FH 定速)
MNET_STAMODE_UD(加減速)	MNET_STAMODE_FL(FL 定速) MNET_STAMODE_FH(FH 定速) MNET_STAMODE_UD(加減速)

上記以外のスタートモード指定時、あるいは速度が未設定の場合は「速度データ無効エラー」となります。 なお、補間関数実行により当該軸の速度は未設定状態となりますので、補間関数実行後に非補間の動作を起動する場合は改めて MNET6045SetSpeed 関数で速度設定を行う必要があります。

8−2−5. 移動データ構造体

移動データの設定に使用します。

書式

[Visual C++]

```
struct TMNET6045Dst {
  long Pos[4];                  // 目標位置/移動量
  long Center[4];               // 円弧中心
  short Direction[4];           // 動作方向
  short PreReg;                 // プリレジスタ書込
  short StartMode;              // スタートモード
};
```

[Visual Basic]

```
Type TMNET6045Dst
  Pos(0 to 3) As Long           '/ 目標位置/移動量
  Center(0 to 3) As Long        '/ 円弧中心
  Direction(0 to 3) As Integer  '/ 動作方向
  PreReg As Integer             '/ プリレジスタ書込
  StartMode As Integer          '/ スタートモード
End Type
```

[VB.NET]

```
<StructLayout(LayoutKind.Sequential)> Structure TMNET6045Dst
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim Pos() As Integer '/ 目標位置/移動量
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim Center() As Integer '/ 動作方向
  <MarshalAs(UnmanagedType.ByValArray, sizeconst:=4)> Dim Direction() As Short '/ 動作方向
  Dim PreReg As Short        '/ プリレジスタ書込
  Dim StartMode As Short     '/ スタートモード (FL/FH/UD)

  Public Sub Initialize()
      ReDim Pos(3)
      ReDim Center(3)
      ReDim Direction(3)
  End Sub
End Structure
```

[メンバー]

メンバー	内容	設定の必要性	設定値
Pos[4]	目標位置	目標位置または移動量設定時	INC命令の場合 現在値からのINC量として扱われます
Center[4]	円弧中心	円弧移動時	INC命令の場合 現在値からのINC量として扱われます
Direction[4]	動作方向	円弧、Jog、原点復帰、原点抜出、原点サーチ、ELまで、EL抜出の方向指定移動時	0 : +方向, CW 1 : -方向, CCW
PreReg	次動作書込	動作指令時	0 : 通常書込 1 : 次動作書込
StartMode	スタートモード	動作指令時	50h : FL定速スタート 51h : FH定速スタート 53h : 加減速スタート

・メンバーに[4]の付くデータは配列メンバーです。データは各軸ごとに設定します。
・対象軸以外への設定は不要です。（対象軸以外の設定は無視されます(動作に影響しません)）
・円弧補間時の動作方向は対象軸に関わらずDirection[0](配列番号:0)に設定ください。

設定上の注意

Visual C++での使用の場合

構造体定義時にメンバーを下記のように初期化(0クリア)してください。

```
TMNET6045Dst dst = {0};   // 構造体定義と初期化(0クリア)
```

初期化を行わない場合メンバーに残った不定データにより正しい動作が行われない可能性があります。

VB.NETでの使用の場合

構造体定義時にメンバーを下記のように初期化してください。

```
Dim dst As New TMNET6045Dst   ’移動データ構造体
dst.Initialize()              ’構造体初期化
```

8−2−6. 加減速設定

8−2−6−1. 加減速指定方法

加減速データ設定時、加減速度を加速(減速)時間で指定するか、加速(減速)レート(1秒あたりの増分(減分)パルス)で指定するかを、速度データ構造体のメンバー「AdcSpc」に設定します。　この指定に従い、メンバー「Acc」，「Dec」に設定された値は次のように扱われます。

識別子	値	内容
MNET_ADC_BY_RATE	0	レート指定(1秒あたりの増分(減分)パルス)
MNET_ADC_BY_TIME	1	時間指定　(秒)

8−2−6−2. 加減速設定の自動補正

設定された加減速データはDLL内部で計算され、LSI：PCL6045Bのレジスタに設定されますが、計算結果がレジスタ長を超える場合や設定可能最小値: 1に満たない場合に設定可能最大値/最小値に自動補正する機能があります。　この機能はデフォルトで有効ですが、無効にしたい場合はMNET6045SetDLLPar関数を使用し設定できます。

識別子	値	内容	設定範囲	設定値／取得値 (網掛け：デフォルト)
MNET6045_DLLPAR_ADCCR_DIS	5E2h	加減速設定自動補正無効	0/1	0:補正する 1:補正しない

[補正する場合]

最大/最小値を超える場合は最大/最小値に補正されます。

[補正しない場合]

最大/最小値を超える場合は加速レート計算結果エラー/減速レート計算結果エラーになります。

8－2－7. Ｓ字動作設定

Ｓ字動作設定は速度データ構造体メンバー「SMode」で行います。

SMode

0：Ｓ字無し

1：Ｓ字有り

Ｓ字範囲は下図のように設定されます。

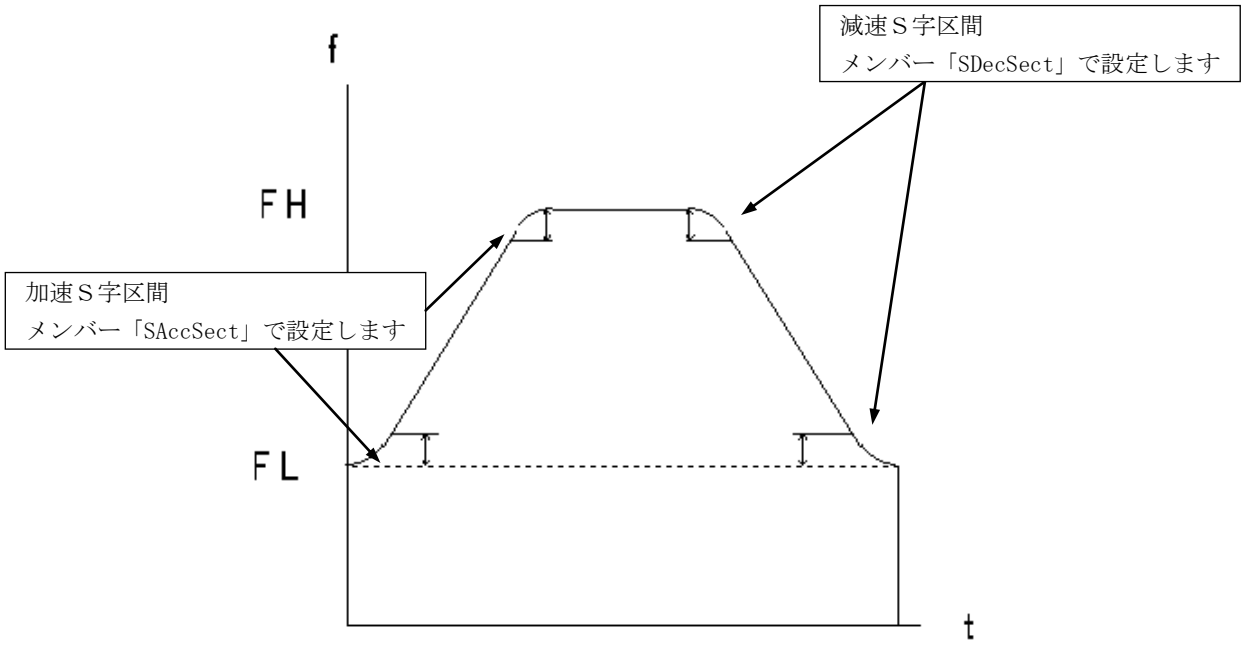
加速時における、Ｓ字動作の範囲はメンバー「SAccSect」で行い、Ｓ字動作の範囲をパルス速度で指定します。

減速時における、Ｓ字動作の範囲はメンバー「SDecSect」で行い、Ｓ字動作の範囲をパルス速度で指定します。

Ｓ字有り（SMode：1）で加速Ｓ字範囲(「SAccSect」)に0を設定した場合、自動的に(FH-FL)/2の値が適用され、加速全域にわたってＳ字動作となり、直線加速部分のない加速となります。

同様にＳ字有り（SMode：1）で減速Ｓ字範囲(「SDecSect」)に0を設定した場合、自動的に(FH-FL)/2の値が適用され、減速全域にわたってＳ字動作となり、直線減速部分のない減速となります。

また(FH-FL)/2より大きい値の設定はエラーとなります。



移動データ構造体メンバーの設定と、対応する加減速動作を下表にまとめます。

加減速動作	「SMode」設定	「SAccSect」,「SDecSect」設定
直線加減速(全域が直線）注１	0	直線加減速となるため設定は無効です。
Ｓ字加減速(全域がＳ字）注１	1	0
直線部分のあるＳ字加減速	1	0以上（FH-FL)/2 以下の範囲で設定します。注２

注１．全域とはあくまで加減速動作域についてとなります。

注２．(FH-FL)/2を超える設定はＳ字範囲設定エラーとなります。

8−2−8. 原点復帰モード

原点復帰動作はMNET6045Org関数で行います。
原点復帰動作の設定はMNET6045SetPar関数で行います。
選択する原点復帰モードを以下に示します。

原点復帰においての注意など

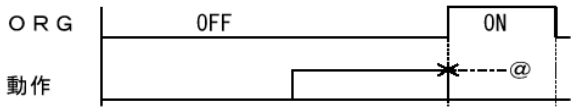
① 原点復帰動作においてもMNET6045SetPar関数で設定したソフトリミットは有効となります。
② ORG信号を使用する原点復帰動作ではORG入力のOFF→ONにより処理されます。
既にORG入力がONしている場合、あるいはORG信号ゼンサーを越えた位置にいる場合、設定した方向に動き続けることになりますので注意ください。
③ 原点復帰動作によりCOUNTER1～COUNTER4がリセットされます。
動作によってカウンタのリセット位置が異なりますのでご注意ください。

原点復帰シーケンス図内の記号について

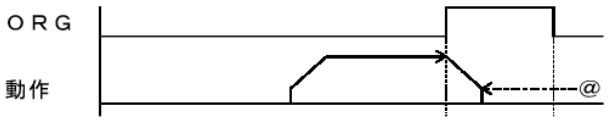
@ : 原点復帰停止時のERC信号出力のタイミング(原点復帰時にERC信号を出力する設定の場合)
＊ : 以下の設定である場合、この位置でもERC信号が出力されます。
・ エラー停止時ERC出力：「出力する」 かつ
・ EL入力時処理:「即停止」

8−2−8−1. 原点復帰モード:0

FL定速動作の場合



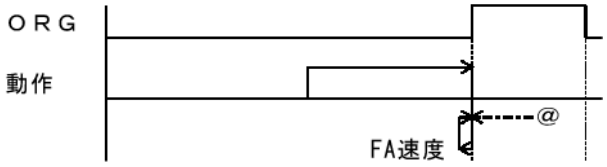
加減速動作の場合



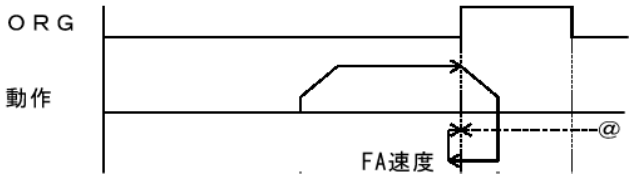
・ORG 入力OFF →ON で即停止(加減速動作時は減速停止)する。
・COUNTER ﾘｾｯﾄﾀｲﾐﾝｸﾞ：ORG 入力OFF →ON の時。

8−2−8−2. 原点復帰モード：1

FL定速動作の場合



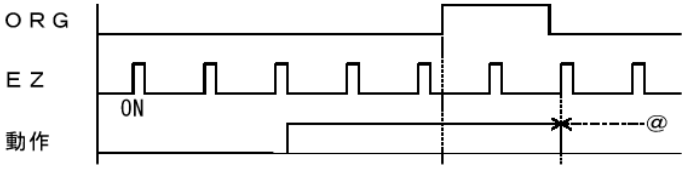
加減速動作の場合



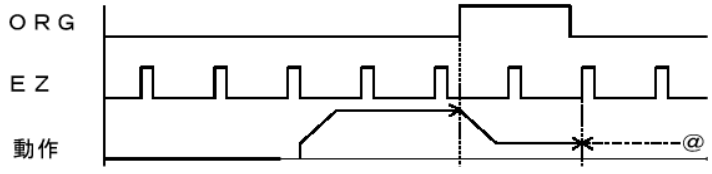
・ORG 入力OFF →ON で即停止(加減速動作時は減速停止)後、FA速度で逆方向へORG 入力ON →OFF まで動作し、その後FA速度で初めの方向へ動作しORG 入力OFF →ON で即停止する。
・COUNTER ﾘｾｯﾄﾀｲﾐﾝｸﾞ：ORG 入力OFF →ON の時。

8−2−8−3. 原点復帰モード：2

FL定速動作の場合



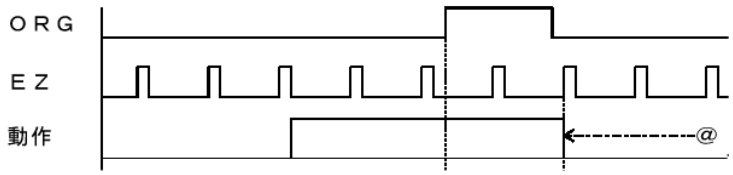
加減速動作の場合



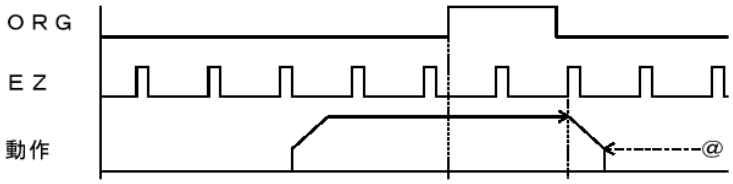
・FL動作時はORG 入力OFF →ON 後のEZｶｳﾝﾄｱｯﾌﾟで即停止。
・加減速動作時はORG 入力OFF →ON で減速してEZｶｳﾝﾄｱｯﾌﾟで即停止する。
・COUNTER ﾘｾｯﾄﾀｲﾐﾝｸﾞ：EZｶｳﾝﾄｱｯﾌﾟ時。

8−2−8−4. 原点復帰モード：3

FL定速動作の場合



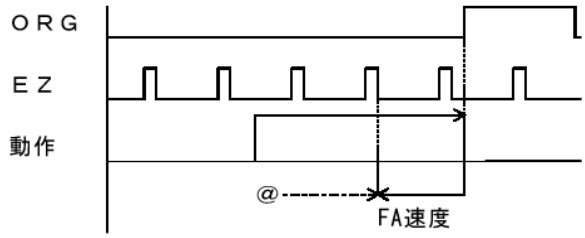
加減速動作の場合



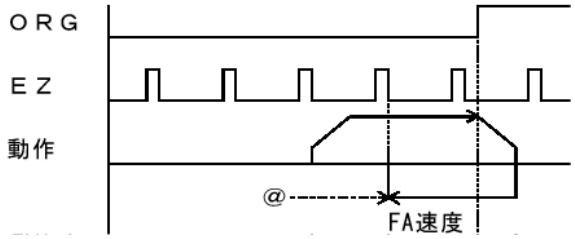
・FL動作時はORG 入力OFF →ON 後のEZｶｳﾝﾄｱｯﾌﾟで即停止。
・加減速動作時はORG 入力OFF →ON 後のEZｶｳﾝﾄｱｯﾌﾟで減速停止。
・COUNTER ﾘｾｯﾄﾀｲﾐﾝｸﾞ：EZｶｳﾝﾄｱｯﾌﾟ時。

8−2−8−5. 原点復帰モード：4

FL定速動作の場合



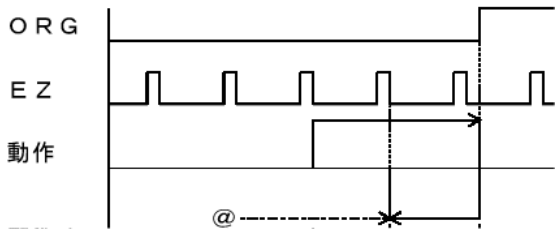
加減速動作の場合



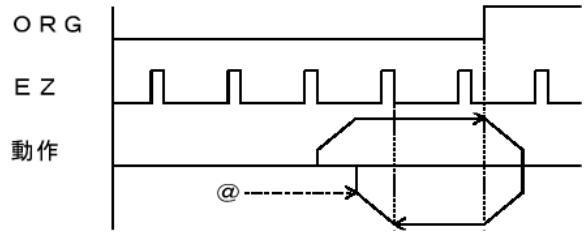
・ORG 入力OFF →ON で即停止(加減速動作時は減速停止)後にFA速度で逆転し、ORG 入力ON →OFF後のEZｶｳﾝﾄｱｯﾌﾟ時に即停止する。
・COUNTER ﾘｾｯﾄﾀｲﾐﾝｸﾞ：EZｶｳﾝﾄｱｯﾌﾟ時。

8−2−8−6. 原点復帰モード:5

FL定速動作の場合



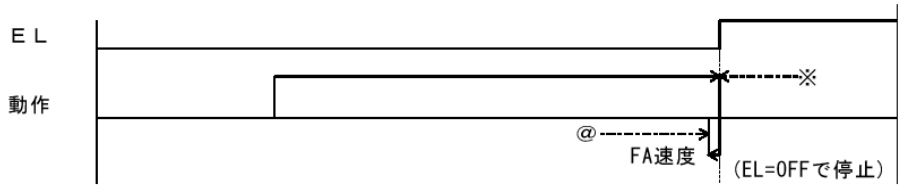
加減速動作の場合



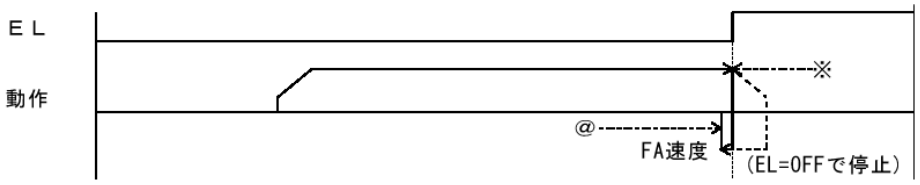
・ORG 入力OFF →ON で即停止(加減速動作時は減速停止)後に指定したｽﾀｰﾄﾓｰﾄﾞで逆転し、
ORG 入力ON →OFF 後のEZｶｳﾝﾄｱｯﾌﾟ時に即停止(加減速動作時は減速停止)。
・COUNTER ﾘｾｯﾄﾀｲﾐﾝｸﾞ：EZｶｳﾝﾄｱｯﾌﾟ時。

8−2−8−7. 原点復帰モード:6

FL定速動作の場合



加減速動作の場合

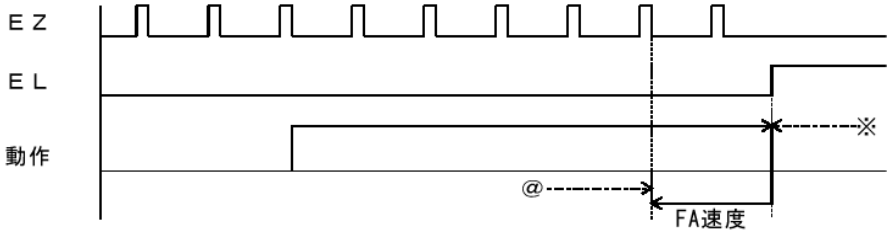


・EL入力ON で即停止(EL入力時処理"減速停止"の時は減速停止)後、FA速度で逆転し、EL入力OFFで即停止する。
・COUNTER ﾘｾｯﾄﾀｲﾐﾝｸﾞ：EL入力OFF時。

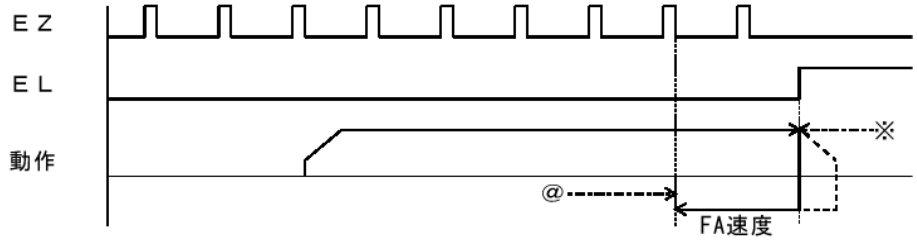
Motionnet®
Remote I/O & Remote Motion

8−2−8−8. 原点復帰モード:7

FL定速動作の場合



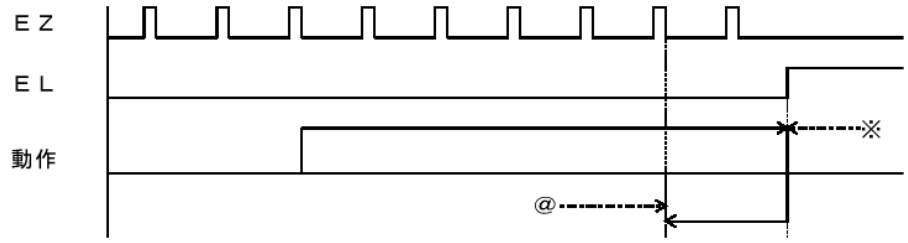
加減速動作の場合



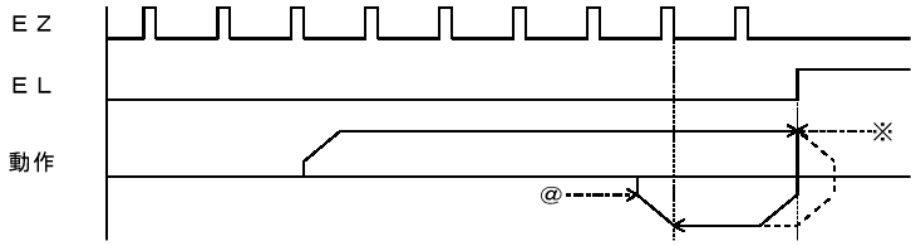
・EL入力ONで即停止(EL入力時処理"減速停止"の時は減速停止)後、FA速度で逆転し、EL入力OFF 後のEZｶｳﾝﾄｱｯﾌﾟ時に即停止する。
・COUNTER ﾘｾｯﾄﾀｲﾐﾝｸﾞ：EZｶｳﾝﾄｱｯﾌﾟで即停止時。

8−2−8−9. 原点復帰モード:8

FL定速動作の場合



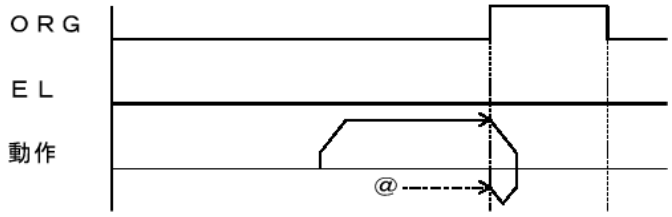
加減速動作の場合



・EL入力ON で即停止(EL入力時処理"減速停止"の時は減速停止)後に指定したｽﾀｰﾄﾓｰﾄﾞで逆転し、EL入力OFF 後のEZｶｳﾝﾄｱｯﾌﾟ時に、即停止(FH動作時は減速停止)。
・COUNTER ﾘｾｯﾄﾀｲﾐﾝｸﾞ：EZｶｳﾝﾄｱｯﾌﾟ時。

8−2−8−10. 原点復帰モード:９

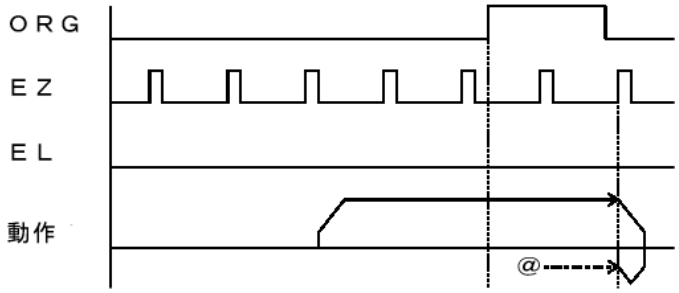
加減速動作の場合



・原点復帰動作０ の動作後、０点復帰(COUNTER2=0 まで動作)する。

8−2−8−11. 原点復帰モード:１０

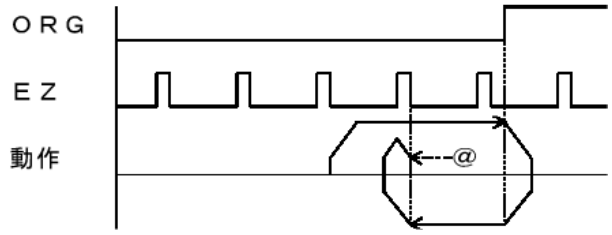
加減速動作の場合



・原点復帰動作３ の動作後、０点復帰(COUNTER2=0 まで動作)する。

8−2−8−12. 原点復帰モード:１１

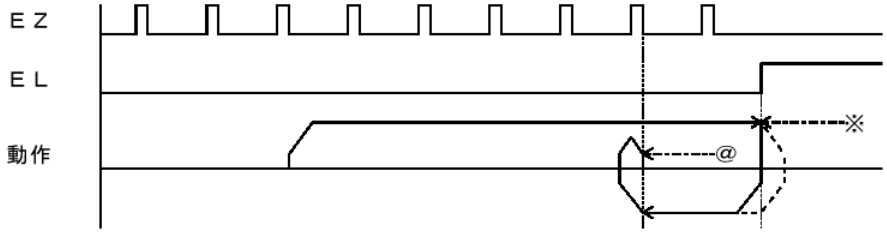
加減速動作の場合



・原点復帰動作５ の動作後、０点復帰(COUNTER2=0 まで動作)する。

8－2－8－13. 原点復帰モード：１２

加減速動作の場合



・原点復帰動作８ の動作後、０点復帰(COUNTER2=0 まで動作)する。

8−2−9. 原点サーチ動作

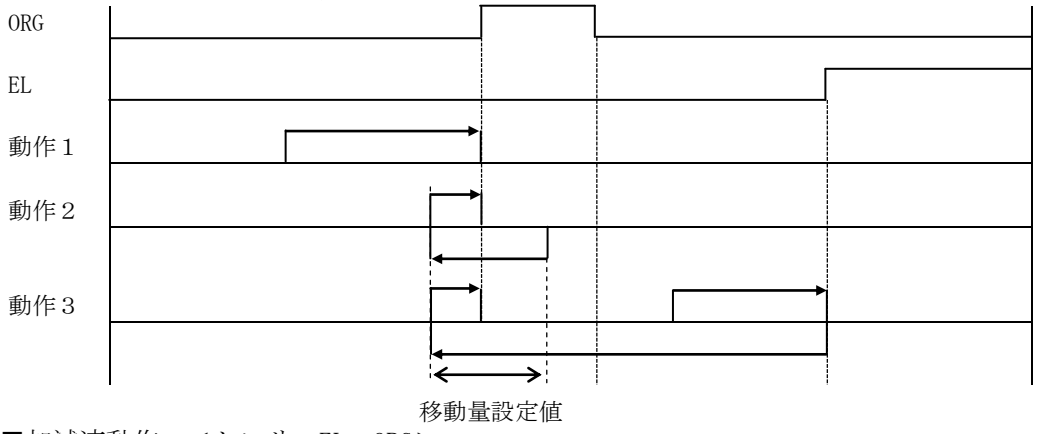
原点サーチ動作とは原点復帰動作に機能を追加したモードで、下記の動作ブロックで構成されます。

①指定と逆方向の「原点復帰動作」
②指定と逆方向の「位置決め動作による原点抜け出し動作」
③指定方向の「原点復帰動作」

原点サーチ動作の起動はMNET6045SearchOrg関数で行います。
原点復帰モードはMNET6045SetPar関数で設定したモードとなります。(原点復帰モード:6,7,8,12では本関数は利用できません)

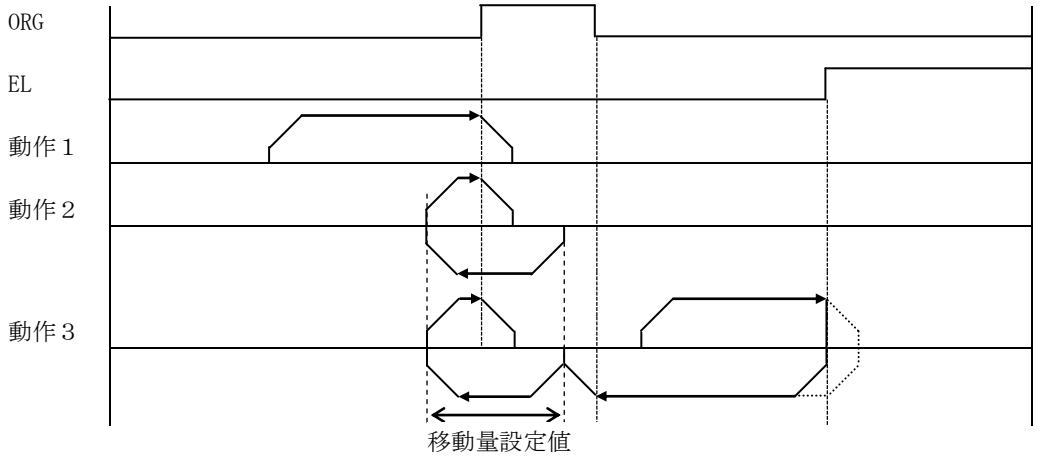
[原点復帰モード:０の例]

□定速動作　＜センサ：EL, ORG＞



■加減速動作　＜センサ：EL, ORG＞

正常停止時でも原点位置ではありませんが、COUNTER2(機械位置)は信頼できる値となります。



動作１　：スタート後ORG 入力がON で停止した時は正常停止になります。
動作２　：スタート時に既にORG 入力がON している場合には、「位置決め動作による原点抜け出し動作」を行った後で、「原点復帰動作」を行います。
動作３　：指定方向に動作してEL で停止した場合には、逆方向への「原点復帰動作」と「位置決め動作による原点抜け出し動作」を行った後で、再度、指定方向の「原点復帰動作」を行います。尚、「位置決め動作による原点抜け出し動作」では、MNET6045SearchOrg関数で設定した移動量の位置決め動作が、原点を抜け出すまで繰り返し行われます。

8－2－10. 動作前確認

本DLLではモーション関数実行前の状態確認を行うか/省略するかはパラメータになっており、MNET6045SetDLLPar関数を使用し設定できます。

識別子	値	内容	設定範囲	設定値／取得値 (網掛け：デフォルト)
MNET6045_DLLPAR_PRECHK_DIS	5E1h	軸動作前確認省略	0/1	0:省略しない,1:省略

[省略しない場合]

「デバイス番号」、「軸番号/軸パターン」、「動作中でないこと」、「動作方向のEL入力」、「ALM入力」、「EMG入力」、「目的の信号入力状態(ORG,ELなど)」、「次動作書込みの場合、書き込み可能かどうか」の確認が行われます。

[省略する場合]

「デバイス番号」、「軸番号/軸パターン」、「動作中でないこと」、「次動作書込みの場合、書き込み可能かどうか」の確認が行われますが、その他の入出力信号や指令値の確認などは行われません。
ALMなどの入力がある場合、動作指令は発行後エラーとなり、エラー割込ありビット(SERR)がONすることになります。　エラー割込要因はMNET6045GetINTFactor関数で確認してください。

Motionnet®
Remote I/O & Remote Motion

8−2−11. 速度設定に関する注意事項

8−2−11−1. 速度倍率

FL/FH速度設定範囲は0.1～6.5MPPSとなっておりますが、DLL内部においては指定値に基づき計算を行い、LSI：PCL6045Bの「FL/FH速度レジスタ」と「速度倍率レジスタ」にデータを設定しております。レジスタ設定値と出力パルス速度の関係は以下のようになっております。

「出力パルス速度」 = 「FL/FH速度レジスタ設定値」×「速度倍率」

例を挙げると速度倍率が10倍、出力パルス速度が150,000PPSである場合

150,000(出力パルス速度) = 15,000(速度レジスタ設定値) × 10倍(速度倍率) となります。

この倍率という考え方により速度レジスタのビット長の制限を越えた速度設定を行えるようになっています。 具体的には各速度設定レジスタ設定範囲は1～65,535であり、それに0.1～100の倍率を乗ずることで0.1～6.5MPPSという速度設定範囲を実現しています。

各速度倍率とその出力パルス速度の範囲を下表に示します。

倍率	出力速度範囲(PPS)	倍率	出力速度範囲(PPS)
0.1	0.1 ～ 6,553.5	5	5 ～ 327,675
0.2	0.2 ～ 13,107.0	10	10 ～ 655,350
0.5	0.5 ～ 32,767.5	20	20 ～ 1,310,700
1	1 ～ 65,535	50	50 ～ 3,276,750
2	2 ～ 131,070	100	100 ～ 6,553,500

倍率はパルス出力中に可変するものではなく、動作スタート前に設定された値に固定となります。

それぞれの倍率は表の通りの範囲を持ち、一つの動作の中では範囲を超えることはできません。

本DLLでは速度データが設定されるたび、 速度データ構造体メンバー 「SpdL」,「SpdH」に設定されたデータの中で、大きい方の値を速度倍率選択速度とし、その速度倍率選択速度を満たす最も小さい速度倍率が自動的に選択され、各速度レジスタの値が計算/設定されています。(小さい速度倍率のほうが設定できる速度の速度間隔が細かくなるため)

基本的には自動的に選択される速度倍率は意識する必要はありませんが、以下の場合に注意が必要です。

設定速度データの丸めについて

各速度レジスタへの設定値は整数値です。

(レジスタ設定値(整数値) = 各SpdL/SpdH指定値(実数値) ÷ 速度倍率(実数値))

したがってSpdL/SpdHに指定された値は丸められる場合があります。

またSpdL/SpdHに指定した値が0に丸められた場合、設定速度は選択された倍率の最小値に自動的に補正されます。

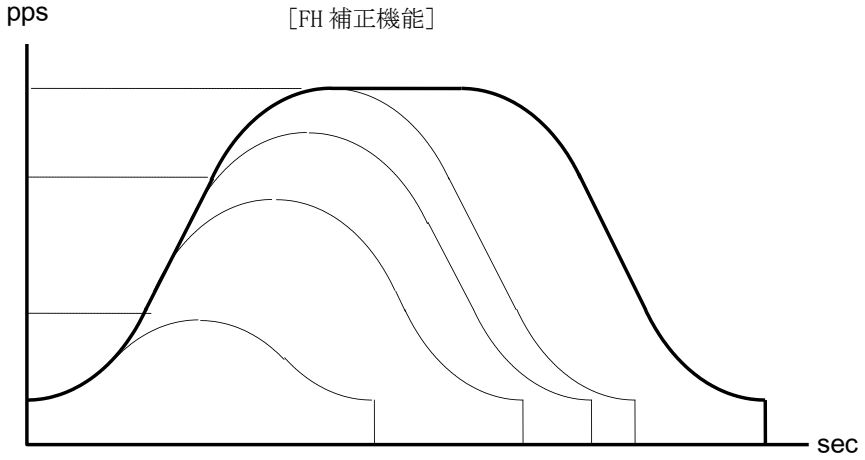
速度オーバーライドを行う場合

通常速度倍率は指定した速度の中で最も大きい速度を基準に自動的に選択されますが、速度オーバーライドを行う場合は予め当該動作スタート前に新速度を満たすことのできる速度倍率を指定して固定する必要があります。

速度倍率の固定指定はMNET6045FixSpeedRange関数で行なわれ、MNET6045UnFixSpeedRange関数で固定指定が解除されるまで指定した速度倍率が適用されます。指定した速度が速度倍率の上限値を超える場合はエラーとなります。

8−2−11−2. FH補正

FH 補正とは位置決め動作が加減速で行なわれる際に、移動量が少ないために設定した FH 速度に到達できず三角駆動となる場合、自動的に FH 速度を低下させて三角駆動を回避する機能です。



移動量による最高速度の自動修正

FH 補正機能の有効/無効は MNET6045SetPar 関数の項目「FH 補正有効/無効」で設定します。

識別子	値	内容	設定範囲	設定値／取得値 (網掛け：デフォルト)
MNET6045_PAR_FHCR_DIS	5C8h	FH補正	0/1	0:有効, 1:無効

注意

減速時間が加速時間の 2 倍を越える速度設定の場合、本パラメータの設定にかかわらず、FH 補正が行われます。

8−2−12. SD入力

SD信号の入力によりFH速度で動作中の軸をFL速度に減速することができます。

SD 信号入力による減速を行う場合はあらかじめ動作起動時にスタートモードに「加減速」を指定し FL速度および加減速データの設定を行っておく必要があります。

（スタートモード「FL 定速」，「FH 定速」で速度データ設定を行った場合、加減速データは設定されません。

SD信号入力の有効/無効、信号論理、他のパラメータはMNET6045SetPar関数で設定します。

SD入力処理に関連するパラメータは以下の通りです。

識別子	値	内容	設定範囲	設定値／取得値(網掛け:デフォルト)
MNET6045_PAR_SD_ENA	5C0h	SD入力(注1)	0/1	0:無効, 1:有効
MNET6045_PAR_SDLTC_ENA	51Ah	SDラッチ有効(注2)	0/1	0:無効, 1:有効
MNET6045_PAR_SD_LOGIC	511h	SD信号論理	0/1	0:負論理, 1:正論理

(注1)SD有効/無効の設定は各動作起動時に設定されます。動作中の移動には設定は適用されません。

(注2)SDラッチが無効である場合、SD入力中のみFL速度までスローダウンし、SD入力がOFFすると、FH速度まで再加速します。SDラッチが有効である場合、動作停止までSD入力はラッチされSD入力がOFFしても再加速しません。

8−2−13. PCS入力

PCS信号の入力によりPCS信号がONした位置を基準にした位置決めができます。

PCS信号入力の有効/無効、信号論理、他のパラメータはMNET6045SetPar関数で設定します。

PCS入力処理に関連するパラメータは以下の通りです。

識別子	値	内容	設定範囲	設定値／取得値(網掛け:デフォルト)
MNET6045_PAR_PCS_ENA	5C2h	PCS入力(注1)	0/1	0:無効, 1:有効
MNET6045_PAR_PCS_LOGIC	515h	PCS信号論理	0/1	0:負論理, 1:正論理

(注1)PCS有効/無効の設定は各動作起動時に設定されます。動作中の移動には設定は適用されません。

PCS動作を行う手順

① MNET6045SetPar関数でPCS信号を「有効」に設定します。

② MNET6045PtpInc関数にPCS入力ONからの位置決め量をセットし実行します。
指定した速度で軸が動作し、PCS入力がONになった地点から位置決めが行われます。

③ MNET6045SetPar関数でPCS信号を「無効」に設定します。

またMNET6045SoftPCS関数を使用することで、PCS入力を代行するコマンドを発行できます。

注意

PCS動作を行なうことのできる動作起動関数はMNET6045PtpIncのみとなります。

またPCS信号有効の場合、次動作書込みは行えません。

Motionnet®
Remote I/O & Remote Motion

8−2−14. 同期スタート

同期スタートとは、軸をスタート待ち状態にし、同時スタート(CSTA)信号や、設定した(LSI)内部同期信号、あるいは指定した軸の停止などによりスタートさせる動作です。

同期スタートに関連するパラメータは MNET6045SetPar 関数で設定します。

同期スタートに関連するパラメータは以下の通りです。

識別子	値	内容	設定範囲	設定値／取得値 (網掛け：デフォルト)
MNET6045_PAR_MSY_MODE	5C4h	同期スタートモード選択 (注1)(注2)	0～3	0:無効 1:同時スタート(CSTA)信号待ち 2:内部同期スタート信号待ち 3:指定軸の停止待ち
MNET6045_PAR_CSTA_TYPE	5B0h	CSTA入力仕様	0/1	0:レベルトリガ, 1:エッジトリガ
MNET6045_PAR_SYNC_OUT_MODE	5B2h	内部同期信号出力タイミング選択(注2)	0～15	1:コンパレータ1条件成立時 2:コンパレータ2条件成立時 8:加速開始時 9:加速終了時 10:減速開始時 11:減速終了時 その他:内部同期信号出力OFF
MNET6045_PAR_SYNC_INP	5B3h	内部同期信号スタート時入力選択(注2)	0～3	0:軸0が出力した内部同期信号 1:軸1が出力した内部同期信号 2:軸2が出力した内部同期信号 3:軸3が出力した内部同期信号
MNET6045_PAR_MSY_SAX	5C5h	他軸停止スタート設定 (注1)(注2)	0～15	停止確認する軸パターンを4ビットで設定します。 ビット0:軸0の停止を確認します ビット1:軸1の停止を確認します ビット2:軸2の停止を確認します ビット3:軸3の停止を確認します デフォルト:0h

(注 1)これらの設定は各動作起動時に設定されます。動作中の移動には設定は適用されません。

(注 2) 補間指令時の設定については以下の制限があります。

①「同期スタートモード選択」設定は補間を行う各軸で同一である必要があります。

②「同期スタートモード選択」が「内部同期スタート信号待ち」の場合は「内部同期信号スタート時入力選択」設定も補間を行なう各軸で同一である必要があります。

③「同期スタートモード選択」が「指定軸の停止待ち」の場合は「他軸停止スタート設定」も補間を行なう各軸で同一である必要があります。

同時スタート(CSTA)信号による動作を行う手順

①MNET6045SetPar 関数で「同時スタートモード選択」を「同時スタート(CSTA)信号待ち」に設定します。

②CSTA設定が「有効」の軸は移動命令を実行すると、同時スタート(CSTA)信号待ち状態となります。(軸の状態はMNET6045GetBitStruct関数などで取得可能です)
同時スタート(CSTA)信号により、動作が開始されます。

(③ MNET6045SetPar 関数で「同期スタートモード選択」を「無効」に設定します。)

同期スタート待ち状態の解除

同期スタート待ち状態の解除はMNET6045Stop関数を停止モード:「即停止」で実行することで行われます。

8－2－15. 同時停止

同時停止は、同時停止(CSTP)信号の入力により複数軸を同時停止させる機能です。
CSTP信号の有効無効および、信号論理などのパラメータはMNET6045SetPar関数で設定します。

CSTP入力処理に関連するパラメータは以下の通りです。

識別子	値	内容	設定範囲	設定値／取得値(網掛け：デフォルト)
MNET6045_PAR_CSTP_ENA	5C6h	CSTP入力	0/1	0:無効, 1:有効
MNET6045_PAR_AUTO_CSTP_ENA	5C7h	CSTP自動出力有効	0/1	0:無効, 1:有効 (異常停止時にCSTP自動出力)
MNET6045_PAR_CSTP_PROC	5B1h	CSTP入力時処理	0/1	0:即停止, 1:減速停止

同時ストップ動作を行う手順

① MNET6045SetPar 関数で CSTP を入力「有効」に設定します。)
② CSTP設定が「有効」の軸は、動作中に同時ストップ(CSTP)信号入力がONした場合、設定した入力時処理で停止します。
(③ MNET6045SetPar 関数で CSTP 入力を「無効」に設定します。)

8－2－16. 基本固定パラメータ

本DLLを操作する場合に、設定が必要な基本固定パラメータの設定内容は以下の通りです。ボードのリセット後、関数ご利用前に必ずこの基本固定パラメータを設定ください。

識別子	値	内容	設定範囲	設定値
MNET6045_PAR_BASE_DATA	500h	基本固定パラメータ設定	0	0を設定ください

設定される内容

① プリレジスタデータがある場合に各動作完了での割込なし
② 円弧補間時終点引き込み有効
③ パルスデューティ 50%
④ 汎用入出力設定 P1,P2:出力,P0,P3～P7:入力
⑤ 原点復帰完了でカウンタ 1～4 リセット
⑥ 入力フィルタ,パルサーフィルタ,エンコーダフィルタ有効
⑦ 指定軸停止同期スタート設定において自軸指定も有効

8−2−17. コンパレータ

各軸毎にコンパレータが 5 本用意されており、カウンタ値などとの比較を行うことができます。
各コンパレータは比較カウンタ、比較方法と条件成立時の処理方法を設定できます。

コンパレータの利用により以下の動作(処理)などを行うことができます。

◆イベント割込を発生
◆動作の即停止／減速停止
◆コンパレータ 1, 2 を利用した、ソフトリミット機能
◆リングカウント機能

コンパレータに関連したパラメータは以下の通りです。

識別子	値	内容	設定範囲	設定値／取得値 (網掛け:デフォルト(リセット時))
MNET6045_PAR_CMP1_CNT	570h	CMP1 比較カウンタ	0～3	0:カウンタ1 1:カウンタ2 2:カウンタ3 3:カウンタ4
MNET6045_PAR_CMP1_MODE	571h	CMP1 比較方法	0～7	0:コンパレータ機能OFF 1:CMP1=比較ｶｳﾝﾀ (ｶｳﾝﾄ方向無関係) 2:CMP1=比較カウンタ (ｶｳﾝﾄｱｯﾌﾟ中) 3:CMP1=比較カウンタ (ｶｳﾝﾄﾀﾞｳﾝ中) 4:CMP1>比較カウンタ 5:CMP1<比較カウンタ 6:+ELとして利用(CMP1 < カウンタ1) 7:コンパレータ機能OFF
MNET6045_PAR_CMP1_PROC	572h	CMP1 成立処理	0～3	0:処理なし 1:即停止 2:減速停止
MNET6045_PAR_CMP1_DATA	573h	コンパレータ 1		コンパレータ1データ 0
MNET6045_PAR_CMP1_RING	574h	CNT1 リングカウント	0/1	0:通常,1:リングカウント
MNET6045_PAR_CMP2_CNT	578h	CMP2 比較カウンタ	0～3	0:カウンタ1 1:カウンタ2 2:カウンタ3 3:カウンタ4
MNET6045_PAR_CMP2_MODE	579h	CMP2 比較方法	0～7	0:コンパレータ機能OFF 1:CMP2=比較ｶｳﾝﾀ (ｶｳﾝﾄ方向無関係) 2:CMP2=比較カウンタ (ｶｳﾝﾄｱｯﾌﾟ中) 3:CMP2=比較カウンタ (ｶｳﾝﾄﾀﾞｳﾝ中) 4:CMP2>比較カウンタ 5:CMP2<比較カウンタ 6:-ELとして利用(CMP2 > カウンタ1) 7:コンパレータ機能OFF
MNET6045_PAR_CMP2_PROC	57Ah	CMP2 成立処理	0～3	0:処理なし 1:即停止 2:減速停止
MNET6045_PAR_CMP2_DATA	57Bh	コンパレータ 2		コンパレータ2データ 0
MNET6045_PAR_CMP2_RING	57Ch	CNT2 リングカウント	0/1	0:通常,1:リングカウント

識別子	値	内容	設定範囲	設定値／取得値 (網掛け:デフォルト(リセット時))
MNET6045_PAR_CMP3_CNT	580h	CMP3 比較カウンタ	0～3	0:カウンタ1 1:カウンタ2 2:カウンタ3 3:カウンタ4
MNET6045_PAR_CMP3_MODE	581h	CMP3 比較方法	0～7	0:コンパレータ機能OFF 1:CMP3=比較ｶｳﾝﾀ (ｶｳﾝﾄ方向無関係) 2:CMP3=比較カウンタ (ｶｳﾝﾄｱｯﾌﾟ中) 3:CMP3=比較カウンタ (ｶｳﾝﾄﾀﾞｳﾝ中) 4:CMP3>比較カウンタ 5:CMP3<比較カウンタ 6:設定禁止 7:コンパレータ機能OFF
MNET6045_PAR_CMP3_PROC	582h	CMP3 成立処理	0～3	0:処理なし 1:即停止 2:減速停止
MNET6045_PAR_CMP3_DATA	583h	コンパレータ 3		コンパレータ3データ 0
MNET6045_PAR_CMP4_CNT	588h	CMP4 比較カウンタ	0～3	0:カウンタ1 1:カウンタ2 2:カウンタ3 3:カウンタ4
MNET6045_PAR_CMP4_MODE	589h	CMP4 比較方法	0～15	0:コンパレータ機能OFF 1:CMP4=比較ｶｳﾝﾀ (ｶｳﾝﾄ方向無関係) 2:CMP4=比較カウンタ (ｶｳﾝﾄｱｯﾌﾟ中) 3:CMP4=比較カウンタ (ｶｳﾝﾄﾀﾞｳﾝ中) 4:CMP4>比較カウンタ 5:CMP4<比較カウンタ 6:-ELとして利用(CMP2 > カウンタ1) 8:IDX(同期)信号出力 (ｶｳﾝﾄ方向無関係) 9:IDX(同期)信号出力 (ｶｳﾝﾄｱｯﾌﾟ中) 10:IDX(同期)信号出力 (ｶｳﾝﾄﾀﾞｳﾝ中) 他:コンパレータ機能OFF
MNET6045_PAR_CMP4_PROC	58Ah	CMP4 成立処理	0～3	0:処理なし 1:即停止 2:減速停止
MNET6045_PAR_CMP4_DATA	58Bh	コンパレータ 4		コンパレータ4データ 0
MNET6045_PAR_CMP4_IDX	58Ch	同期信号出力	0/1	0:CNT=CMP2の間IDX信号を出力 1:CNT4がｶｳﾝﾄにより0になった時に IDX信号を出力

Motionnet®
Remote I/O & Remote Motion

識別子	値	内容	設定範囲	設定値／取得値 (網掛け：デフォルト(リセット時))
MNET6045_PAR_CMP5_CNT	590h	CMP5 比較カウンタ	0～5	0:カウンタ1 1:カウンタ2 2:カウンタ3 3:カウンタ4 4:位置決めカウンタ 5:現在速度
MNET6045_PAR_CMP5_MODE	591h	CMP5 比較方法	0～7	0:コンパレータ機能OFF 1:CMP5=比較ｶｳﾝﾀ (ｶｳﾝﾄ方向無関係) 2:CMP5=比較カウンタ (ｶｳﾝﾄｱｯﾌﾟ中) 3:CMP5=比較カウンタ (ｶｳﾝﾄﾀﾞｳﾝ中) 4:CMP5>比較カウンタ 5:CMP5<比較カウンタ 他:コンパレータ機能OFF
MNET6045_PAR_CMP5_PROC	592h	CMP5 成立処理	0～3	0:処理なし 1:即停止 2:減速停止
MNET6045_PAR_CMP5_DATA	593h	コンパレータ 5		コンパレータ5データ 0

8−2−17−1. ソフトリミットの設定

コンパレータ 1 と 2 の利用によりソフトリミット機能を設定できます。
ソフトリミット動作時にはパルス出力を即(減速)停止させることができます。
また、スタートコマンド書き込み時にソフトリミットが ON 状態の場合、ソフトリミットが ON 状態の方向へはスタートは出来ません。逆方向へはスタートできます。

以下のデータの設定はすべて MNET6045SetPar 関数で行います。

ソフトリミット機能を有効にする手順

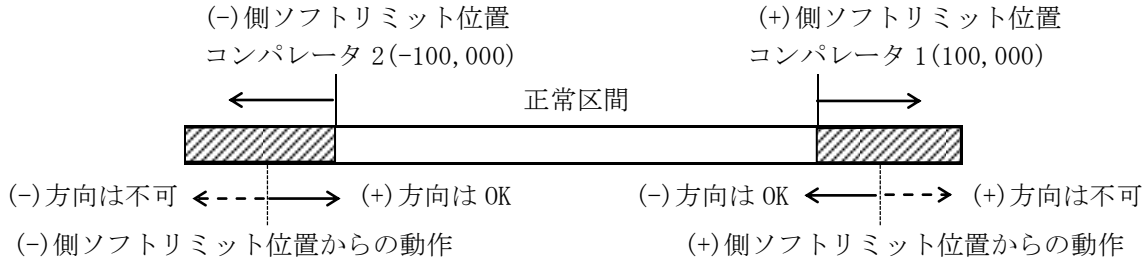
①コンパレータ値の設定
(コンパレータ 1 に+ソフトリミットとする値を、コンパレータ 2 に-ソフトリミットとする値を設定します。)
②コンパレータ比較カウンタの設定
(コンパレータ 1,コンパレータ 2 共に「カウンタ 1」を比較カウンタに設定します。)
③コンパレータ成立時処理の設定
(ソフトエンドリミット到達時(コンパレータ 1,コンパレータ 2 成立時)に行う処理(即停止/減速停止)を設定します。)
④コンパレータ比較方法の設定
(コンパレータ 1,コンパレータ 2 の比較方法に「ソフトエンドリミット」(設定値:6)を設定します。)

ソフトリミット機能を無効にする手順

コンパレータ比較方法の設定
(コンパレータ 1,コンパレータ 2 の比較方法に「コンパレータ機能 OFF」(設定値:0)を設定します。)

設定例

コンパレータ 1 設定値 = 100,000 :(+)側リミット値
コンパレータ 2 設定値 = -100,000 :(−)側リミット値



記述例

[Visual C++]

```
MNET6045SetPar(0, 0, 0, MNET6045_PAR_CMP1_DATA, 100000);   // +SEL 値設定
MNET6045SetPar(0, 0, 0, MNET6045_PAR_CMP2_DATA, -100000); // -SEL 値設定
MNET6045SetPar(0, 0, 0, MNET6045_PAR_CMP1_CNT, 0);         // CMP1 比較カウンタ設定(CNT1)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_CMP2_CNT, 0);         // CMP2 比較カウンタ設定(CNT1)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_CMP1_PROC, 1);        // CMP1 成立時処理(即停止)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_CMP2_PROC, 1);        // CMP2 成立時処理(即停止)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_CMP1_MODE, 6);   // CMP1 比較方法(ソフトエンドリミット)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_CMP2_MODE, 6);   // CMP2 比較方法(ソフトエンドリミット)
```

[Visual Basic]

```
MNET6045SetPar 0, 0, 0, MNET6045_PAR_CMP1_DATA, 100000    '/ +SEL 値設定
MNET6045SetPar 0, 0, 0, MNET6045_PAR_CMP2_DATA, -100000   '/ -SEL 値設定
MNET6045SetPar 0, 0, 0, MNET6045_PAR_CMP1_CNT, 0          '/ CMP1 比較カウンタ設定(CNT1)
MNET6045SetPar 0, 0, 0, MNET6045_PAR_CMP2_CNT, 0          '/ CMP2 比較カウンタ設定(CNT1)
MNET6045SetPar 0, 0, 0, MNET6045_PAR_CMP1_PROC, 1         '/ CMP1 成立時処理(即停止)
MNET6045SetPar 0, 0, 0, MNET6045_PAR_CMP2_PROC, 1         '/ CMP2 成立時処理(即停止)
MNET6045SetPar 0, 0, 0, MNET6045_PAR_CMP1_MODE, 6     '/ CMP1 比較方法(ソフトエンドリミット)
MNET6045SetPar 0, 0, 0, MNET6045_PAR_CMP2_MODE, 6     '/ CMP2 比較方法(ソフトエンドリミット)
```

[VB.NET]

```
MNET6045SetPar(0, 0, 0, MNET6045_PAR_CMP1_DATA, 100000)    '/ +SEL 値設定
MNET6045SetPar(0, 0, 0, MNET6045_PAR_CMP2_DATA, -100000)   '/ -SEL 値設定
MNET6045SetPar(0, 0, 0, MNET6045_PAR_CMP1_CNT, 0)          '/ CMP1 比較カウンタ設定(CNT1)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_CMP2_CNT, 0)          '/ CMP2 比較カウンタ設定(CNT1)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_CMP1_PROC, 1)         '/ CMP1 成立時処理(即停止)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_CMP2_PROC, 1)         '/ CMP2 成立時処理(即停止)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_CMP1_MODE, 6)     '/ CMP1 比較方法(ソフトエンドリミット)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_CMP2_MODE, 6)     '/ CMP2 比較方法(ソフトエンドリミット)
```

ソフトリミット機能利用上の注意

設定したソフトリミットは原点復帰動作中も機能します。

8－2－18. イベント割込設定

イベント割込要因の設定は MNET6045SetPar 関数で行います。
イベント割込を発生させたい内容に対応するビットを１にしたデータを設定ください。

Bit	ﾋﾞｯﾄ名	内容
0	IREN	自動停止時
1	IRN	次動作継続ｽﾀｰﾄ時
2	IRNM	動作用2ndﾌﾟﾘﾚｼﾞｽﾀ書き込み可能時
3	IRND	ｺﾝﾊﾟﾚｰﾀ用2ndﾌﾟﾘﾚｼﾞｽﾀ書き込み可能時
4	IRUS	加速開始時
5	IRUE	加速終了時
6	IRDS	減速開始時
7	IRDE	減速終了時
8	IRC1	ｺﾝﾊﾟﾚｰﾀ1条件成立時
9	IRC2	ｺﾝﾊﾟﾚｰﾀ2条件成立時
10	IRC3	ｺﾝﾊﾟﾚｰﾀ3条件成立時
11	IRC4	ｺﾝﾊﾟﾚｰﾀ4条件成立時
12	IRC5	ｺﾝﾊﾟﾚｰﾀ5条件成立時
13	IRCL	CLR信号入力によるｶｳﾝﾄ値のﾘｾｯﾄ時
14	IRLT	LTC信号入力によるｶｳﾝﾄ値のﾗｯﾁ時
15	IROL	ORG信号入力によるｶｳﾝﾄ値のﾗｯﾁ時
16	IRSD	SD入力ON時
17	IRDR	±DR入力変化時（本ﾎﾞｰﾄﾞでは未使用です）
18	IRSA	CSTA入力ON時
19	未定義	常に0を設定してください
20	未定義	常に0を設定してください
21	未定義	常に0を設定してください
22	未定義	常に0を設定してください
23	未定義	常に0を設定してください
24	未定義	常に0を設定してください
25	未定義	常に0を設定してください
26	未定義	常に0を設定してください
27	未定義	常に0を設定してください
28	未定義	常に0を設定してください
29	未定義	常に0を設定してください
30	未定義	常に0を設定してください
31	未定義	常に0を設定してください

イベント割込発生の有無は MNET6045GetBit 関数または MNET6045GetBitStruct 関数で確認できます。
イベント割込要因の読出しは MNET6045GetINTFactor 関数で行います。
イベント割込要因データは MNET6045GetINTFactor 関数実行によりリセットされ 0 になりますのでご注意ください。

8−2−19. ラッチ

各カウンタは下記のタイミングで値をラッチでき、ラッチした値の読出しは MNET6045GetLtc 関数で行うことができます。

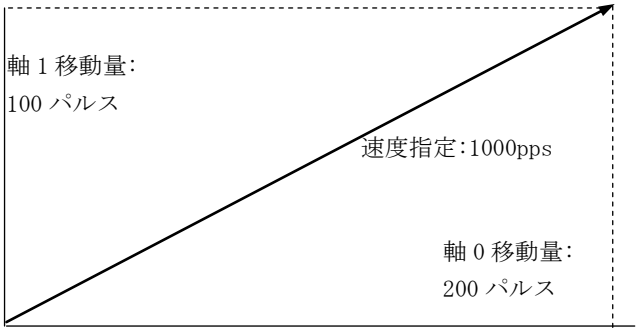
① ORG 入力時
② コンパレータ 4 条件成立時
③ コンパレータ 5 条件成立時
④ MNET6045SoftLtc 関数の実行(ラッチ入力代行コマンドの書込み)

ラッチに関連するパラメータは以下の通りです。

識別子	値	内容	設定範囲	設定値／取得値 (網掛け：デフォルト)
MNET6045_PAR_LTC_MODE	598h	ラッチタイミング	0～3	0:LTC入力時（注1） 1:ORG入力時 2:CMP4成立時 3:CMP5成立時
MNET6045_PAR_SPD_LTC	599h	ラッチ3でカウンタ3の代わりに速度データをラッチします	0/1	0:無効, 1:有効
MNET6045_PAR_LTC_INP_DIS	59Ah	LTC入力無効	0/1	0:有効, 1:無効
MNET6045_PAR_CNT1_LTC_RST	59Bh	ラッチ時カウンタ1リセット	0/1	0:無効, 1:有効
MNET6045_PAR_CNT2_LTC_RST	59Ch	ラッチ時カウンタ2リセット	0/1	0:無効, 1:有効
MNET6045_PAR_CNT3_LTC_RST	59Dh	ラッチ時カウンタ3リセット	0/1	0:無効, 1:有効
MNET6045_PAR_CNT4_LTC_RST	59Eh	ラッチ時カウンタ4リセット	0/1	0:無効, 1:有効

8－2－20. 直線補間

MNET6045Line 関数、MNET6045LineInc 関数の利用により直線補間動作を行うことができます。
動作速度の設定は軸パターンに関わらず、速度データ構造体の軸 0 のデータに設定します。
設定した速度データは全補間軸間のベクトル速度(合成速度)となります。



この場合、各軸の速度はそれぞれおよそ、軸 0 が 894.4pps, 軸 1 が 447.2pps となります。

直線補間を行うにあたり、以下の注意点があります。

(1) 同期スタートを行う場合、補間に含まれる軸の同期スタート設定を同一にしておく必要があります。
(2) 軸パターンに含まれる軸が 1 軸だけの場合、あるいは補間軸全てが移動量 0 の場合はエラーとなります。
(3) 次動作設定で現動作に連続して直線補間動作を行わせる際に、対象軸の現動作が同時終了とならない Ptp 動作などの場合、 同時にスタートさせるには現動作を補間動作とするか、「同期スタート(他軸の停止によるスタート)」をご利用ください。
(4) 合成速度一定の設定は補間軸全てで同一である必要があります。
(5) 合成速度一定の場合、加速データと減速データが同一である必要があります。
(6) 直線補間動作関数の実行により MNET6045SetSpeed 関数で設定した非補間動作用の速度データは未設定状態となります。

8－2－21. 円弧補間

MNET6045Arc 関数、MNET6045ArcInc 関数の利用により円弧補間動作を行うことができます。
動作速度の設定は軸パターンに関わらず、速度データ構造体の軸 0 のデータに設定します。

円弧補間を行うにあたり、以下の注意点があります。

(1) 同期スタートを行う場合、補間に含まれる軸の同期スタート設定を同一にしておく必要があります。
(2) 軸パターンに含まれる軸が 2 軸でない場合はエラーとなります。
(3) 次動作設定で現動作に連続して円弧補間動作を行わせる際に、対象軸の現動作が同時終了とならない Ptp 動作などの場合、 同時にスタートさせるには現動作を補間動作とするか、「同期スタート(他軸の停止によるスタート)」をご利用ください。
(4) 合成速度一定の設定は補間軸全てで同一である必要があります。
(5) 合成速度一定の場合、加速データと減速データが同一である必要があります。
(6) 円弧補間動作関数の実行により MNET6045SetSpeed 関数で設定した非補間動作用の速度データは未設定状態となります。

8−2−22. 次動作設定

本ボード上のLSI:PCL6045Bには、現在動作中の動作の２つ先までを書き込み、現在動作中の動作が完了した時点で自動的に次の動作をスタートさせる機能があります。この次動作書き込みのためのレジスタをプリレジスタと呼びます。

プリレジスタへの次動作の書き込みは移動データ構造体のメンバー「PreReg」に「1」を設定して行います。また、プリレジスタの空き状態はMNET6045GetBitまたはMNET6045GetBitStruct関数で取得できます。

次動作書込みを行う上での注意など

① プリレジスタ満杯時の指令はエラーとなります。(動作中の動作には影響ありません)
② エラー割込要因にデータが残っている場合、次動作書き込みは受け付けられません。
③ 現動作が最終到達位置の確定しない(MNET6045Continue,ORG,ELなどの)動作のうしろには次動作は設定できません。

次動作のキャンセル

次動作のキャンセルはMNET6045CancelPreReg関数,またはMNET6045Stop関数で行われます。

8−2−23. 速度オーバーライド

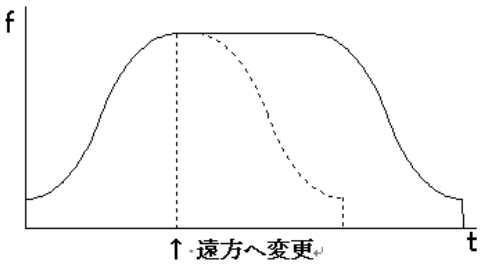
加減速動作の場合、MNET6045OvrideSpd関数で動作中の速度変更を行うことができます。
ただし、位置決め動作で残りパルスが少ない場合や減速中での速度変更は受け付けられません。

・速度変更時の加減速度は動作スタート時に設定した加減速設定となります。
(速度設定時の加減速指定方法が「時間」であっても、起動時に計算された加減速の傾きが適用されます)
・オーバーライドできる速度は動作起動時の速度倍率によります。したがって予めオーバーライドされる速度の最大値を含む速度倍率に固定設定しておく必要があります。
・指定した軸が直線補間中である場合、指定した速度は合成速度として扱われオーバーライドが行なわれます。
・指定した軸が円弧補間中である場合、指定した速度は円弧補間の速度となります。

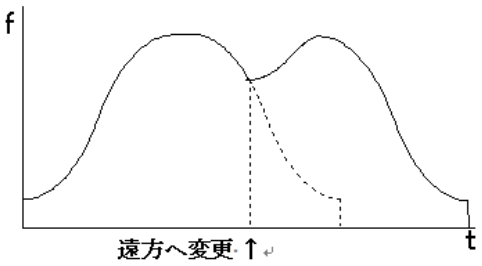
8−2−24. 位置オーバーライド

MNET6045OvridePos 関数を使用することで、目標位置のオーバーライドができます。
目標位置のオーバーライドが可能な動作は Ptp および PtpInc 動作のみとなります。

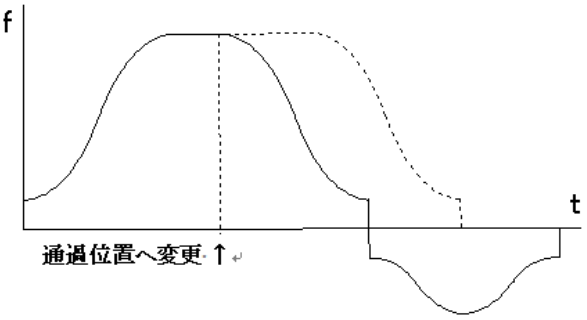
加速・定速動作中に新データを最初の目標位置より
遠方に変更した場合、そのままの速度パターンで動作し
新データ位置で位置決め完了になります。



減速中に新データを最初の目標位置より遠方に
変更した場合、その位置から FH 速度に再加速後、
新データ位置で位置決め完了になります。



新データ位置をすでに通過している場合、
または減速中で最初の目標位置より手前に
変更した場合、減速停止後に逆転して新データ位置
で位置決め完了になります。



加速／減速動作を行うのは加減速スタート時のみです。また、目標位置のオーバーライドは、位置決め動作完了まで何度でも可能です。

位置のオーバーライドは、動作中(FL/FH 定速動作中,加/減速中,バックラッシュ補正動作)にだけ有効です。
停止直前にオーバーライドを行った場合、オーバーライドが受け付けられない場合があります。
停止直前にオーバーライドを行う可能性がある場合、オーバーライドの成否判定は停止時の位置により判断して下さい。　あるいはオーバーライドに失敗した場合はエラー割込が発生し、エラー割込要因レジスタの対応するビットが ON します。したがって MNET6045GetINTFactor 関数でのエラー割込要因の読出しでも行うことができます。

8－2－25. パルサー同期動作

パルサー同期動作とはPA/PB信号の入力に同期してパルスを発信する動作です。
パルサー同期動作機能は以下の通り用意されています。

関数	機能
MNET6045PulserPtpInc	パルサー入力に応じ補間無し位置決め(INC)動作をします
MNET6045PulserPtp	パルサー入力に応じ補間無し位置決め(ABS)動作をします
MNET6045PulserLineInc	パルサー入力に応じ直線補間(INC)動作をします
MNET6045PulserLine	パルサー入力に応じ直線補間(ABS)動作をします
MNET6045PulserArcInc	パルサー入力に応じ円弧補間(INC)動作をします
MNET6045PulserArc	パルサー入力に応じ円弧補間(ABS)動作をします
MNET6045PulserContinue	パルサー入力に応じ連続動作します
MNET6045PulserPtpIncNo	パルサー入力に応じ補間無し位置決め(INC)動作をします(軸番号指定)
MNET6045PulserPtpNo	パルサー入力に応じ補間無し位置決め(ABS)動作をします(軸番号指定)
MNET6045PulserContinueNo	パルサー入力に応じ連続動作します(軸番号指定)

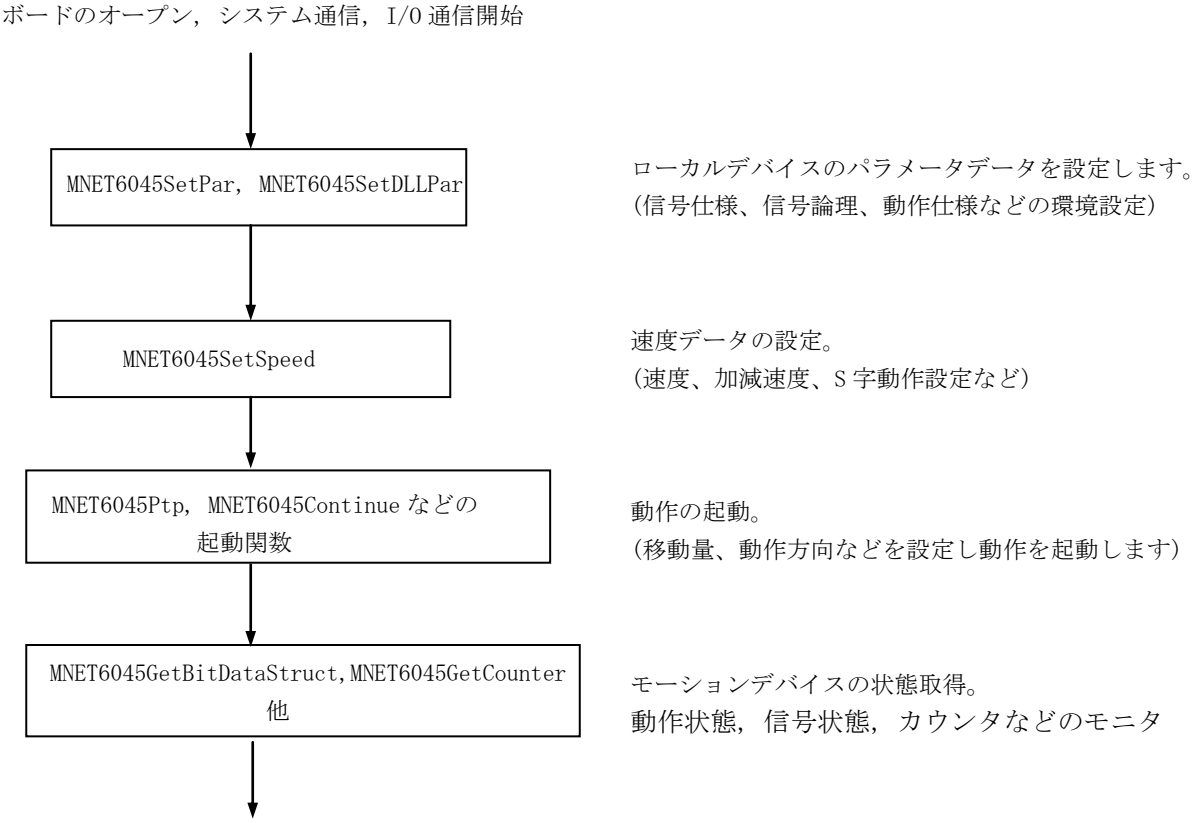
注意

パルサー同期動作は FL 定速動作のみで御利用いただけます。

8－2－26. 操作の流れ

8－2－26－1. 操作フロー

システム起動から４軸(6045)モーションデバイスを操作する流れの概略は次のようになります。



8−2−26−2. プログラム記述例

[Visual C++]

```
    TMNET6045Spd spd = {0};     // 速度データ構造体
    TMNET6045Dst dst = {0};     // 移動データ構造体
    TMNET6045BitData bit = {0}; // ビットデータ構造体
    short axPtn;              // 軸パターン(軸 3～0 を 4 ビットで表現したデータ)
    short stopSts;            // 位置決め完了確認用
    bool posEnd = false;      // 位置決め完了状態

    PPCI_L112Open();              // ボードのオープン
    MNETCenterReset(0);           // センターデバイスリセット
    MNETSysComm(0);               // システム通信
    MNETIOCommStart(0);           // I/O 通信スタート

    // パラメータ設定　(以下は一部の項目の設定例です)
    MNET6045SetPar(0, 0, 0, MNET6045_PAR_BASE_DATA, 0);     // ボード固定値
    MNET6045SetPar(0, 0, 0, MNET6045_PAR_TYPE_OUTPLS, 4);  // 出力パルス仕様(CW/CCW 負論理)
    MNET6045SetPar(0, 0, 0, MNET6045_PAR_ENA_DIR_TIM, 0);  // 方向変化タイマー(有効)
    MNET6045SetPar(0, 0, 0, MNET6045_PAR_LOGIC_SD, 1);     // SD 信号論理(正論理)
    MNET6045SetPar(0, 0, 0, MNET6045_PAR_LOGIC_ORG, 1);    // ORG 信号論理(正論理)
    MNET6045SetPar(0, 0, 0, MNET6045_PAR_LOGIC_ALM, 0);    // ALM 信号論理(負論理)
    MNET6045SetPar(0, 0, 0, MNET6045_PAR_LOGIC_ERC, 1);    // ERC 信号論理(正論理)
    MNET6045SetPar(0, 0, 0, MNET6045_PAR_LOGIC_INP, 1);    // INP 信号論理(正論理)
    MNET6045SetPar(0, 0, 0, MNET6045_PAR_LOGIC_PCS, 1);    // PCS 信号論理(正論理)
    MNET6045SetPar(0, 0, 0, MNET6045_PAR_MODE_EL, 0);      // EL 入力時処理(即停止)
    MNET6045SetPar(0, 0, 0, MNET6045_PAR_MODE_ALM, 0);     // ALM 入力時処理(即停止)

    // 速度データ構造体データ設定
    spd.SpdL[0] = 100;                                      // FL 速度設定
    spd.SpdH[0] = 5000;                                     // FH 速度設定
    spd.Acc[0] = 1.5;                                       // 加速データ設定
    spd.Dec[0] = 1.5;                                       // 減速データ設定
    spd.AdcSpc[0] =MNET_ADC_BY_TIME;                        // 加減速指定方法:時間
    spd.SMode[0] =MNET_NON_S;                               // S 字なし
    MNET6045SetSpeed(0, 0, 0x1, &spd, MNET_STAMODE_UD);    // 速度データ設定

    // 移動データ構造体データ設定
    dst.Pos[0] = 100000;                                    // 移動量データ設定
    dst.StartMode = MNET_STAMODE_UD;                        // スタートモード設定

    MNET6045PtpInc(0, 0, 0x1, &dst);                        // INC 指定 Ptp 移動

    while (stopSts == 0) {
      MNET6045StopChk(0, 0, 0x1, &stopSts);                 // 軸停止状態確認
    }

    MNETCenterReset(0);                                     // センターデバイスリセット
    PPCI_L112Close();                                       // ボードのクローズ
```


Motionnet®
Remote I/O & Remote Motion

[Visual Basic]

```
    Dim spd As TMNET6045Spd       ’速度データ構造体
    Dim dst As TMNET6045Dst       ’移動データ構造体
    Dim bit As TMNET6045BitData   ’ビットデータ構造体
    Dim axPtn As Integer          ’軸パターン(軸3～0を4ビットで表現したデータ)
    Dim stopSts As Integer        ’位置決め完了確認用
    Dim posEnd As Boolean         ’位置決め完了状態

    PPCI_L112Open                 ’ボードのオープン
    MNETCenterReset 0             ’センターデバイスリセット
    MNETSysComm 0                 ’システム通信
    MNETIOCommStart 0             ’I/O通信スタート

    ’パラメータ設定　(以下は一部の項目の設定例です)
    MNET6045SetPar 0, 0, 0, MNET6045_PAR_BASE_DATA, 0       ’ボード固定値
    MNET6045SetPar 0, 0, 0, MNET6045_PAR_TYPE_OUTPLS, 4     ’出力パルス仕様(CW/CCW負論理)
    MNET6045SetPar 0, 0, 0, MNET6045_PAR_ENA_DIR_TIM, 0     ’方向変化タイマー(有効)
    MNET6045SetPar 0, 0, 0, MNET6045_PAR_LOGIC_SD, 1        ’SD信号論理(正論理)
    MNET6045SetPar 0, 0, 0, MNET6045_PAR_LOGIC_ORG, 1       ’ORG信号論理(正論理)
    MNET6045SetPar 0, 0, 0, MNET6045_PAR_LOGIC_ALM, 0       ’ALM信号論理(負論理)
    MNET6045SetPar 0, 0, 0, MNET6045_PAR_LOGIC_ERC, 1       ’ERC信号論理(正論理)
    MNET6045SetPar 0, 0, 0, MNET6045_PAR_LOGIC_INP, 1       ’INP信号論理(正論理)
    MNET6045SetPar 0, 0, 0, MNET6045_PAR_LOGIC_PCS, 1       ’PCS信号論理(正論理)
    MNET6045SetPar 0, 0, 0, MNET6045_PAR_MODE_EL, 0         ’EL入力時処理(即停止)
    MNET6045SetPar 0, 0, 0, MNET6045_PAR_MODE_ALM, 0        ’ALM入力時処理(即停止)

    ’速度データ構造体データ設定
    spd.SpdL(0) = 100                                       ’FL速度設定
    spd.SpdH(0) = 5000                                      ’FH速度設定
    spd.Acc(0) = 1.5                                        ’加速データ設定
    spd.Dec(0) = 1.5                                        ’減速データ設定
    spd.AdcSpc(0) = MNET_ADC_BY_TIME                        ’加減速指定方法:時間
    spd.SMode(0) = MNET_NON_S                               ’ S字なし
    MNET6045SetSpeed 0, 0, &H1, spd, MNET_STAMODE_UD        ’ 速度データ設定

    ’移動データ構造体データ設定
    dst.Pos(0) = 100000                                     ’移動量データ設定
    dst.StartMode = MNET_STAMODE_UD                         ’スタートモード設定

    MNET6045PtpInc 0, 0, &H1, dst                           ’INC指定Ptp移動

    Do While (stopSts = 0)
      MNET6045StopChk 0, 0, &H1, stopSts                    ’ 軸停止状態確認
    Loop

    MNETCenterReset 0                                       ’センターデバイスリセット
    PPCI_L112Close                                          ’ボードのクローズ
```

[VB.NET]

```
Dim spd As New TMNET6045Spd ’速度データ構造体
Dim dst As New TMNET6045Dst ’移動データ構造体
Dim bit As TMNET6045BitData ’ビットデータ構造体
Dim axPtn As Integer ’軸パターン(軸3～0を4ビットで表現したデータ)
Dim stopSts As Integer ’位置決め完了確認用
Dim posEnd As Boolean ’位置決め完了状態
spd.Initialize() ’構造体初期化
dst.Initialize() ’構造体初期化

PPCI_L112Open() ’ボードのオープン
MNETCenterReset( 0 ) ’センターデバイスリセット
MNETSysComm( 0 ) ’システム通信
MNETIOCommStart( 0 ) ’I/O通信スタート
‘ パラメータ設定 (以下は一部の項目の設定例です)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_BASE_DATA, 0) ’ボード固定値
MNET6045SetPar(0, 0, 0, MNET6045_PAR_TYPE_OUTPLS, 4) ’出力パルス仕様(CW/CCW負論理)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_ENA_DIR_TIM, 0) ’方向変化タイマー(有効)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_LOGIC_SD, 1) ’SD信号論理(正論理)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_LOGIC_ORG, 1) ’ORG信号論理(正論理)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_LOGIC_ALM, 0) ’ALM信号論理(負論理)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_LOGIC_ERC, 1) ’ERC信号論理(正論理)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_LOGIC_INP, 1) ’INP信号論理(正論理)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_LOGIC_PCS, 1) ’PCS信号論理(正論理)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_MODE_EL, 0) ’EL入力時処理(即停止)
MNET6045SetPar(0, 0, 0, MNET6045_PAR_MODE_ALM, 0) ’ALM入力時処理(即停止)

‘ 速度データ構造体データ設定
spd.SpdL(0) = 100 ’FL速度設定
spd.SpdH(0) = 5000 ’FH速度設定
spd.Acc(0) = 1.5 ’加速データ設定
spd.Dec(0) = 1.5 ’減速データ設定
spd.AdcSpc(0) = MNET_ADC_BY_TIME ’加減速指定方法:時間
spd.SMode(0) = MNET_NON_S ’S字なし
MNET6045SetSpeed(0, 0, &H1, spd, MNET_STAMODE_UD) ’速度データ設定

‘ 移動データ構造体データ設定
dst.Pos(0) = 100000 ’移動量データ設定
dst.StartMode = MNET_STAMODE_UD ’スタートモード設定

MNET6045PtpInc(0, 0, &H1, dst) ’INC指定Ptp移動

Do While (stopSts = 0)
  MNET6045StopChk(0, 0, &H1, stopSts) ’軸停止状態確認
Loop

MNETCenterReset( 0 ) ’センターデバイスリセット
PPCI_L112Close() ’ボードのクローズ
```

8−3. データ定義

8−3−1. 構造体定義

8−3−1−1. ビットデータ構造体

ビットデータの読出しに使用し一括で取得することが可能です。(MNET6045GetBitStruct 関数使用時)

書式

[Visual C++]

```
struct TMNET6045BitData {
  short Srun;     // 動作中
  short Send;     // 動作停止
  short Serr;     // エラー割込発生
  short Sint;     // イベント割込発生
  short Scp1;     // CMP1成立
  short Scp2;     // CMP2成立
  short Scp3;     // CMP3成立
  short Scp4;     // CMP4成立
  short Scp5;     // CMP5成立
  short Sprf;     // 次動作用プリレジスタ満杯

  short Mpin1;    // 汎用入力1
  short Mpout1;   // 汎用出力1
  short Mpout2;   // 汎用出力2
  short Sfu;      // 加速中
  short Sfd;      // 減速中
  short Sfc;      // 定速動作中
  short Salm;     // ALM入力
  short Spel;     // PEL入力
  short Smel;     // MEL入力
  short Sorg;     // ORG入力

  short Cnd;      // 動作状態
  short Sdir;     // 動作方向
  short Ssta;     // CSTA入力
  short Sstp;     // CSTP入力
  short Semg;     // EMG入力
  short Spcs;     // PCS入力
  short Serc;     // ERC出力
  short Sez;      // Z相入力
  short Sdin;     // SD入力
  short Sinp;     // INP入力
  short Spfm;     // 次動作用プリレジスタモニタ

  short P3;       // 汎用入出力P3
  short P4;       // 汎用入出力P4
  short P5;       // 汎用入出力P5
  short Sclr;     // CLR入力
  short Sltc;     // LTC入力
};
```

[Visual Basic]

```
Type TMNET6045BitData
  Srun As Integer    '/ 動作中
  Send As Integer    '/ 動作停止
  Serr As Integer    '/ エラー割込発生
  Sint As Integer    '/ イベント割込発生
  Scp1 As Integer    '/ CMP1成立
  Scp2 As Integer    '/ CMP2成立
  Scp3 As Integer    '/ CMP3成立
  Scp4 As Integer    '/ CMP4成立
  Scp5 As Integer    '/ CMP5成立
  Sprf As Integer    '/ 次動作用プリレジスタ満杯

  Mpin1 As Integer   '/ 汎用入力1
  Mpout1 As Integer  '/ 汎用出力1
  Mpout2 As Integer  '/ 汎用出力2
  Sfu As Integer     '/ 加速中
  Sfd As Integer     '/ 減速中
  Sfc As Integer     '/ 定速動作中
  Salm As Integer    '/ ALM入力
  Spel As Integer    '/ PEL入力
  Smel As Integer    '/ MEL入力
  Sorg As Integer    '/ ORG入力

  Cnd As Integer     '/ 動作状態
  Sdir As Integer    '/ 動作方向
  Ssta As Integer    '/ CSTA入力
  Sstp As Integer    '/ CSTP入力
  Semg As Integer    '/ EMG入力
  Spcs As Integer    '/ PCS入力
  Serc As Integer    '/ ERC出力
  Sez As Integer     '/ Z相入力
  Sdin As Integer    '/ SD入力
  Sinp As Integer    '/ INP入力
  Spfm As Integer    '/ 次動作用プリレジスタモニタ

  P3 As Integer      '/ 汎用入出力P3
  P4 As Integer      '/ 汎用入出力P4
  P5 As Integer      '/ 汎用入出力P5
  Sclr As Integer    '/ CLR入力
  Sltc As Integer    '/ LTC入力
End Type
```

[VB.NET]

```
Structure TMNET6045BitData
    Dim Srun As Short   '/ RUN状態
    Dim Send As Short   '/ 動作停止
    Dim Serr As Short   '/ エラー割込発生
    Dim Sint As Short   '/ イベント割込発生
    Dim Scp1 As Short   '/ CMP1成立
    Dim Scp2 As Short   '/ CMP2成立
    Dim Scp3 As Short   '/ CMP3成立
    Dim Scp4 As Short   '/ CMP4成立
    Dim Scp5 As Short   '/ CMP5成立
    Dim Sprf As Short   '/ 次動作用プリレジスタ満杯
    Dim Mpin1 As Short  '/ 汎用入力1
    Dim Mpout1 As Short '/ 汎用出力1
    Dim Mpout2 As Short '/ 汎用出力2
    Dim Sfu As Short    '/ 加速中
    Dim Sfd As Short    '/ 減速中
    Dim Sfc As Short    '/ 定速動作中
    Dim Salm As Short   '/ ALM入力
    Dim Spel As Short   '/ PEL入力
    Dim Smel As Short   '/ MEL入力
    Dim Sorg As Short   '/ ORG入力
    Dim Cnd As Short    '/ 動作状態
    Dim Sdir As Short   '/ 動作方向
    Dim Ssta As Short   '/ CSTA入力
    Dim Sstp As Short   '/ CSTP入力
    Dim Semg As Short   '/ EMG入力
    Dim Spcs As Short   '/ PCS入力
    Dim Serc As Short   '/ ERC出力
    Dim Sez As Short    '/ Z相入力
    Dim Sdin As Short   '/ SD入力
    Dim Sinp As Short   '/ INP入力
    Dim Spfm As Short   '/ 次動作用プリレジスタモニタ
    Dim P3 As Short     '/ 汎用入出力P3
    Dim P4 As Short     '/ 汎用入出力P4
    Dim P5 As Short     '/ 汎用入出力P5
    Dim Sclr As Short   '/ CLR入力
    Dim Sltc As Short   '/ LTC入力
End Structure
```

[メンバー]

メンバー	内容	取得値
Srun	動作中	1:動作中
Send	動作完了	1:動作完了
Serr	エラー割込状態	1:エラー割込有 (REST(エラー割込ステータス)読出しにより0となります)
Sint	イベント割込状態	1:イベント割込有 (RIST(イベント割込ステータス)読出しにより0となります)
Scp1	コンパレータ1成立	1:成立
Scp2	コンパレータ2成立	1:成立
Scp3	コンパレータ3成立	1:成立
Scp4	コンパレータ4成立	1:成立
Scp5	コンパレータ5成立	1:成立
Sprf	プリレジスタ満杯	1:次動作用プリレジスタ満杯(次動作登録不可)
Mpin1	汎用入力1信号状態	0:OFF, 1:ON
Mpout1	汎用出力1信号状態	0:OFF, 1:ON
Mpout2	汎用出力2信号状態	0:OFF, 1:ON
Sfu	加速中	1:加速中
Sfd	減速中	1:減速中
Sfc	定速動作中	1:定速動作中
Salm	alm信号状態	0:OFF, 1:ON
Spel	pel信号状態	0:OFF, 1:ON
Smel	mel信号状態	0:OFF, 1:ON
Sorg	org信号状態	0:OFF, 1:ON
Cnd	軸動作状態	軸動作状態 別表参照
Sdir	動作方向	0:+方向, 1:-方向
Ssta	csta信号状態	0:OFF, 1:ON
Sstp	cstp信号状態	0:OFF, 1:ON
Semg	emg信号状態	0:OFF, 1:ON
Spcs	pcs信号状態	0:OFF, 1:ON
Serc	erc信号状態	0:OFF, 1:ON
Sez	ez信号状態	0:OFF, 1:ON
Sdin	sd信号状態	0:OFF, 1:ON
Sinp	inp信号状態	0:OFF, 1:ON
Spfm	プリレジスタ状態	プリレジスタ使用状態 (0:未動作～3：満杯)
P3	汎用入出力P3状態	0:OFF, 1:ON
P4	汎用入出力P4状態	0:OFF, 1:ON
P5	汎用入出力P5状態	0:OFF, 1:ON
Sclr	clr信号状態	0:OFF, 1:ON
Sltc	ltc信号状態	0:OFF, 1:ON

信号状態については論理パラメータが加味された結果となります。

cnd(軸動作状態)一覧

取得値	状態	取得値	状態
0	停止中	8	PA/PB入力待ち
1	DR入力待ち	9	FA定速動作中
2	CSTA(同時スタート)入力待ち	10	FL定速動作中
3	内部同期信号待ち	11	加速中
4	他軸停止待ち	12	FH定速動作中
5	ERCタイマー完了待ち	13	減速中
6	方向変化タイマー完了待ち	14	INP入力待ち
7	バックラッシュ補正中	15	その他(スタート制御中)

8−3−2. ステータス／ビットマップ

8−3−2−1. エラー割込要因

エラー割込要因のマップです。

エラー割込が発生したとき、対応するビットが 1 になります。

データの取得は MNET6045GetINTFactor 関数で行います。

このデータは LSI:PCL6045B の REST レジスタ(エラー割込ステータス)から取得しています。このレジスタのデータは読出しによりリセットされ 0 となります。

Bit	ﾋﾞｯﾄ名	内容
0	ESC1	コンパレータ1条件成立による停止時(+ソフトリミット)
1	ESC2	コンパレータ2条件成立による停止時(-ソフトリミット)
2	ESC3	コンパレータ3条件成立による停止時
3	ESC4	コンパレータ4条件成立による停止時
4	ESC5	コンパレータ5条件成立による停止時
5	ESPL	+EL入力による停止時
6	ESML	-EL入力による停止時
7	ESAL	ALM入力による停止時
8	ESSP	CSTP入力による停止時
9	ESEM	CEMG入力による停止時
10	ESSD	SD入力による減速停止時
11	未定義	常に0
12	ESDT	補間データ異常による停止時
13	ESIP	補間動作中に他軸の異常停止による同時停止時
14	ESPO	PA/PB入力用バッファカウンタオーバーフロー発生による停止時
15	ESAO	補間動作時の位置決めカウンタのカウント範囲オーバー発生による停止時
16	ESEE	EA/EB入力エラー発生時(停止しません)
17	ESPE	PA/PB入力エラー発生時(停止しません)
18	未定義	常に0
19	未定義	常に0
20	未定義	常に0
21	未定義	常に0
22	未定義	常に0
23	未定義	常に0
24	未定義	常に0
25	未定義	常に0
26	未定義	常に0
27	未定義	常に0
28	未定義	常に0
29	未定義	常に0
30	未定義	常に0
31	未定義	常に0

8−3−2−2. イベント割込要因

イベント割込要因のマップです。

MNET6045SetPar 関数で設定したイベントが発生したとき、対応するビットが 1 になります。

データの取得は MNET6045GetINTFactor 関数で行います。

このデータは LSI:PCL6045B の RIST レジスタ(イベント割込ステータス)から取得しています。このレジスタのデータは読出しによりリセットされ 0 となります。

イベント割込の設定は「8-2-18.イベント割込設定」を参照ください。

Bit	ﾋﾞｯﾄ名	内容
0	ISEN	自動停止時
1	ISN	次動作継続スタート時
2	ISNM	動作用2ndプリレジスタ書き込み可能時
3	ISND	コンパレータ用2ndプリレジスタ書き込み可能時
4	ISUS	加速開始時
5	ISUE	加速終了時
6	ISDS	減速開始時
7	ISDE	減速終了時
8	ISC1	コンパレータ1条件成立時
9	ISC2	コンパレータ2条件成立時
10	ISC3	コンパレータ3条件成立時
11	ISC4	コンパレータ4条件成立時
12	ISC5	コンパレータ5条件成立時
13	ISCL	CLR信号入力によるカウント値のリセット時
14	ISLT	LTC信号入力によるカウント値のラッチ時
15	ISOL	ORG信号入力によるカウント値のラッチ時
16	ISSD	SD入力ON時
17	ISPD	+DR入力変化時（本ボードでは未使用です）
18	ISMD	-DR入力変化時（本ボードでは未使用です）
19	ISSA	CSTA入力ON時
20	未定義	
21	未定義	
22	未定義	
23	未定義	
24	未定義	
25	未定義	
26	未定義	
27	未定義	
28	未定義	
29	未定義	
30	未定義	
31	未定義	

8−3−3. 識別子

8−3−3−1. スタートモード

スタートモードは下記3パターンがあります。

MNET6045SetSpeed関数での速度データ設定時に速度データ構造体のメンバー「StartMode」に、および移動命令関数の引数に設定するデータとその識別子は次のようになります。

識別子	値	内容
MNET_STAMODE_FL	0050h	FL定速
MNET_STAMODE_FH	0051h	FH 定速
MNET_STAMODE_UD	0053h	加減速

8−3−3−2. 動作方向

連続動作や原点復帰、円弧補間動作など、動作方向を指定する移動命令の場合に、移動データ構造体のメンバー「Direction」に設定します。

識別子	値	内容
MNET_DIR_PLUS	0	プラス方向(CW)
MNET_DIR_MINUS	1	マイナス方向(CCW)

8−3−3−3. 加減速指定方法

MNET6045SetSpeed関数などでの加減速データ設定時、加減速度を加速(減速)時間で指定するか、加速(減速)レート(1秒あたりの増分(減分)パルス)で指定するかを、速度データ構造体のメンバー「AdcSpc」に設定します。

識別子	値	内容
MNET_ADC_BY_RATE	0	レート指定(1秒あたりの増分(減分)パルス速度)
MNET_ADC_BY_TIME	1	時間指定　(秒)

8−3−3−4. S字使用/未使用

MNET6045SetSpeed関数などでの速度データ設定時、S字動作を使用するかどうかを、速度データ構造体のメンバー「SMode」に設定します。

識別子	値	内容
MNET_NON_S	0	S字動作なし
MNET_USE_S	1	S字動作

8−3−3−5. カウンタ指定

カウンタデータ取得/設定時に、対象カウンタを指定します。

識別子	値	内容	主な用途
MNET_SEL_COUNTER1	0	カウンタ1選択	指令パルスカウント
MNET_SEL_COUNTER2	1	カウンタ2選択	フィードバックカウント
MNET_SEL_COUNTER3	2	カウンタ3選択	偏差カウント
MNET_SEL_COUNTER4	3	カウンタ4選択	汎用

各カウンタの入力信号の設定は MNET6045SetPar 関数で変更できます。

8−3−3−6. ラッチ指定

ラッチデータ取得時に、対象のラッチを指定します。

識別子	値	内容
MNET_SEL_LTC1	0	ラッチ1選択
MNET_SEL_LTC2	1	ラッチ2選択
MNET_SEL_LTC3	2	ラッチ3選択
MNET_SEL_LTC4	3	ラッチ4選択

8−3−3−7. 停止モード

MNET6045Stop関数使用時に停止方法を指定します。

識別子	値	内容
MNET_STOP	0	即停止
MNET_SDSTOP	1	減速停止
MNET_EMGSTOP	2	非常停止

8−3−3−8. 速度倍率レンジ

MNET6045FixSpeedRange関数使用時に速度倍率を設定します。

識別子	値	内容	速度範囲(pps)
MNET6045_SPD_01_RANGE	0	速度倍率:0.1倍	0.1 ～ 6,553.5
MNET6045_SPD_02_RANGE	1	速度倍率:0.2倍	0.2 ～ 13,107.0
MNET6045_SPD_05_RANGE	2	速度倍率:0.5倍	0.5 ～ 32,767.5
MNET6045_SPD_1_RANGE	3	速度倍率:1倍	1 ～ 65,535
MNET6045_SPD_2_RANGE	4	速度倍率:2倍	2 ～ 131,070
MNET6045_SPD_5_RANGE	5	速度倍率:5倍	5 ～ 327,675
MNET6045_SPD_10_RANGE	6	速度倍率:10倍	10 ～ 655,350
MNET6045_SPD_20_RANGE	7	速度倍率:20倍	20 ～ 1,310,700
MNET6045_SPD_50_RANGE	8	速度倍率:50倍	50 ～ 3,276,750
MNET6045_SPD_100_RANGE	9	速度倍率:100倍	100 ～ 6,553,500

8−3−3−9. ビット選択

MNET6045GetBitData, MNET6045SetBitData関数使用時に対象ビットを指定します。

識別子	値	内容	取得(設定値)
MNET6045_BIT_SRUN	1	SRUNビット状態(動作中)	1:動作中
MNET6045_BIT_SEND	2	SENDビット状態(動作停止)	1:動作停止
MNET6045_BIT_SERR	3	SERRビット状態(エラー割込有)	1:エラー割込有
MNET6045_BIT_SINT	4	SINTビット状態(イベント割込有)	1:イベント割込有
MNET6045_BIT_SCP1	5	CMP1比較条件成立状態	0:非成立, 1:成立
MNET6045_BIT_SCP2	6	CMP2比較条件成立状態	0:非成立, 1:成立
MNET6045_BIT_SCP3	7	CMP2比較条件成立状態	0:非成立, 1:成立
MNET6045_BIT_SCP4	8	CMP2比較条件成立状態	0:非成立, 1:成立
MNET6045_BIT_SCP5	9	CMP2比較条件成立状態	0:非成立, 1:成立
MNET6045_BIT_SPRF	10	プリレジスタ満杯	1:満杯
MNET6045_BIT_MPIN1	20	汎用入力1信号状態	0:OFF, 1:ON
MNET6045_BIT_MPOUT1	21	汎用出力1信号状態	0:OFF, 1:ON **(Set可能)**
MNET6045_BIT_MPOUT2	22	汎用出力2信号状態	0:OFF, 1:ON **(Set可能)**
MNET6045_BIT_SFU	23	加速中	1:加速中
MNET6045_BIT_SFD	24	減速中	1:減速中
MNET6045_BIT_SFC	25	定速動作中	1:定速動作中
MNET6045_BIT_SALM	26	ALM信号状態	0:OFF, 1:ON
MNET6045_BIT_SPEL	27	+EL信号状態	0:OFF, 1:ON
MNET6045_BIT_SMEL	28	-EL信号状態	0:OFF, 1:ON
MNET6045_BIT_SORG	29	ORG信号状態	0:OFF, 1:ON
MNET6045_BIT_CND	40	CND(軸動作状態)	ビットデータ構造体参照
MNET6045_BIT_SDIR	41	動作方向	0:+方向, 1:-方向
MNET6045_BIT_SSTA	42	CSTA信号状態	0:OFF, 1:ON
MNET6045_BIT_SSTP	43	CSTP信号状態	0:OFF, 1:ON
MNET6045_BIT_SEMG	44	EMG信号状態	0:OFF, 1:ON
MNET6045_BIT_SPCS	45	PCS信号状態	0:OFF, 1:ON
MNET6045_BIT_SERC	46	ERC信号状態	0:OFF, 1:ON **(Set可能)**
MNET6045_BIT_SEZ	47	EZ信号状態	0:OFF, 1:ON
MNET6045_BIT_SDIN	48	SD信号状態	0:OFF, 1:ON
MNET6045_BIT_SINP	49	INP信号状態	0:OFF, 1:ON
MNET6045_BIT_SPRM	50	次動作用プリレジスタモニタ	プリレジスタ状態
MNET6045_BIT_P3	60	汎用入出力P3状態	0:OFF, 1:ON **(Set可能)**
MNET6045_BIT_P4	61	汎用入出力P4状態	0:OFF, 1:ON **(Set可能)**
MNET6045_BIT_P5	62	汎用入出力P5状態	0:OFF, 1:ON **(Set可能)**
MNET6045_BIT_SCLR	63	CLR信号状態	0:OFF, 1:ON
MNET6045_BIT_SLTC	64	LTC信号状態	0:OFF, 1:ON

信号状態は論理パラメータが加味された結果となります。
またこの値はLSIとしての値となります。ボードとしての信号論理は各I/F回路を御確認ください。

8−3−3−10. パラメータ項目番号定義

MNET6045GetPar関数, MNET6045SetPar関数使用時に対象のパラメータ項目を指定します。

識別子	値	内容	設定範囲	設定値／取得値 (網掛け:デフォルト(リセット時))
MNET6045_PAR_BASEDATA	500h	基本固定パラメータ(注1)	0	0 を設定ください
MNET6045_PAR_OUTPLS_TYPE	501h	出力パルス仕様	0～7	当項下部記載の出力パルス仕様参照 0
MNET6045_PAR_DIR_TIM_DIS	502h	方向変化タイマ	0/1	0:有効, 1:無効
MNET6045_PAR_IDLE_PLS	503h	アイドリングパルス	0～7	アイドリングパルス 0
MNET6045_PAR_VIBE_SUPPRESS1	504h	振動抑制 1		当項下部記載の振動抑制参照 0
MNET6045_PAR_VIBE_SUPPRESS2	505h	振動抑制 2		当項下部記載の振動抑制参照 0
MNET6045_PAR_ALM_LOGIC	510h	ALM 論理	0/1	0:負論理, 1:正論理
MNET6045_PAR_SD_LOGIC	511h	SD 論理	0/1	0:負論理, 1:正論理
MNET6045_PAR_ORG_LOGIC	512h	ORG 論理	0/1	0:負論理, 1:正論理
MNET6045_PAR_ERC_LOGIC	513h	ERC 論理	0/1	0:負論理, 1:正論理
MNET6045_PAR_INP_LOGIC	514h	INP 論理	0/1	0:負論理, 1:正論理
MNET6045_PAR_PCS_LOGIC	515h	PCS 論理	0/1	0:負論理, 1:正論理
MNET6045_PAR_EZ_LOGIC	516h	Z 相論理	0～7	0:立下り, 1:立上り
MNET6045_PAR_ALM_PROC	517h	ALM 入力時処理	0/1	0:即停止, 1:減速停止 (注 3)
MNET6045_PAR_EL_PROC	518h	EL 入力時処理	0/1	0:即停止, 1:減速停止 (注 2,3)
MNET6045_PAR_SD_PROC	519h	SD 入力時処理	0/1	0:減速, 1:減速停止
MNET6045_PAR_SDLTC_ENA	51Ah	SD ラッチ	0/1	0:無効, 1:有効
MNET6045_PAR_INP_FILT_ENA	51Bh	入力フィルタ	0/1	0:無効, 1:有効
MNET6045_PAR_ERR_ERC_PROC	51Ch	エラー停止時 ERC 出力	0/1	0:しない, 1: 出力する
MNET6045_PAR_ORG_ERC_PROC	51Dh	原点時 ERC 出力	0/1	0:しない, 1: 出力する
MNET6045_PAR_ERCPLS_LEN	51Eh	ERC パルス幅	0～7	0:12μs, 1:102μs, 2:409μs 3:1.6ms 4:13ms, 5:52ms, 6:104ms, 7:ﾚﾍﾞﾙ出力
MNET6045_PAR_CLR_LOGIC	51Fh	CLR 論理	0～3	0:立下りエッジ 1:立上りエッジ 2:L レベル 3:H レベル
MNET6045_PAR_LTC_LOGIC	520h	LTC 論理	0/1	0:立下りエッジ 1:立上りエッジ

注1. 本DLLの仕様により決まっている固定パラメータ設定です。 本DLL使用時には必ず設定ください。設定される内容については「8-2-16.基本固定パラメータ」を参照ください。

注2. EL入力ON時の処理を減速停止に設定した場合、EL入力ONで減速を開始しますので、EL位置を通過して停止します機械系への衝突等に十分御注意下さい。

注3. 停止方法を減速停止に設定した場合には、減速して停止するまで入力ON状態を保持して下さい。正常停止と異常停止との判別は停止タイミングの後で決定されます。そのため、加減速位置決め動作で減速中に異常停止信号が入力されても正常停止と認識する場合があります。その場合でも異常停止信号を入力し続けていれば、次動作は継続せずに異常停止になります。

識別子	値	内容	設定範囲	設定値／取得値 (網掛け:デフォルト(リセット時))
MNET6045_PAR_P3_TYPE	521h	P3 仕様	0～3	0:汎用入力 1:汎用出力 2:CP1(CMP1 成立状態負論理出力) 3:CP1(CMP1 成立状態正論理出力)
MNET6045_PAR_P4_TYPE	522h	P4 仕様	0～3	0:汎用入力 1:汎用出力 2:CP2(CMP2 成立状態負論理出力) 3:CP2(CMP2 成立状態正論理出力)
MNET6045_PAR_P5_TYPE	523h	P5 仕様	0～3	0:汎用入力 1:汎用出力 2:CP3(CMP3 成立状態負論理出力) 3:CP3(CMP3 成立状態正論理出力)
MNET6045_PAR_ORG_MODE	530h	原点復帰方法	0～12	8-2-8 参照 0
MNET6045_PAR_EZ_CNT	531h	Z 相回数	0～15	設定値+1 回の動作となります 0
MNET6045_PAR_ENC_TYPE	540h	エンコーダ入力仕様	0～3	0:90 度位相差 1 逓倍 1:90 度位相差 2 逓倍 2:90 度位相差 4 逓倍 3:EA の立上りでカウントアップ, EB の立上りでカウントダウン
MNET6045_PAR_ENC_REV	541h	エンコーダ入力反転	0/1	0:なし, 1:反転
MNET6045_PAR_ENC_FILT_ENA	542h	エンコーダフィルタ	0/1	0:無効, 1:有効
MNET6045_PAR_ENC_DIS	543h	EA/EB 入力	0/1	0:有効, 1:無効
MNET6045_PAR_PULSER_TYPE	550h	パルサー入力仕様	0～3	0:90 度位相差 1 逓倍 1:90 度位相差 2 逓倍 2:90 度位相差 4 逓倍 3:PA の立上りでカウントアップ, PB の立上りでカウントダウン
MNET6045_PAR_PULSER_REV	551h	パルサー入力反転	0/1	0:無効, 1:有効
MNET6045_PAR_PULSER_FILT_ENA	552h	パルサーフィルタ	0/1	0:無効, 1:有効
MNET6045_PAR_PULSER_DIS	553h	パルサー入力	0/1	0:有効, 1:無効
MNET6045_PAR_PULSER_DIV	554h	パルサー入力分周比	0～2047	設定値/2048に分周されます 0設定時は分周されません。0
MNET6045_PAR_PULSER_MULT	555h	パルサー入力逓倍比	0～31	(設定値+1)倍に逓倍されます。0
MNET6045_PAR_CNT2_MODE	560h	CNT2 入力選択	0～2	0:EA/EB入力 1:出力パルス 2:PA/PB入力
MNET6045_PAR_CNT3_MODE	561h	CNT3 入力選択	0～2	0:出力ﾊﾟﾙｽとEA/EB入力(偏差ｶｳﾝﾄ) 1:出力ﾊﾟﾙｽとPA/PB入力(偏差ｶｳﾝﾄ) 2:EA/EB入力とPA/PB入力(偏差ｶｳﾝﾄ)
MNET6045_PAR_CNT4_MODE	562h	CNT4 入力選択	0～3	0:出力パルス 1:EA/EB入力 2:PA/PB入力 3:CLK(19.6608MHz)の1/2分周クロック
MNET6045_PAR_CNT1_CLR_ENA	565h	CLR入力時CNT1クリア	0/1	0:無効, 1:有効
MNET6045_PAR_CNT2_CLR_ENA	566h	CLR入力時CNT2クリア	0/1	0:無効, 1:有効
MNET6045_PAR_CNT3_CLR_ENA	567h	CLR入力時CNT3クリア	0/1	0:無効, 1:有効
MNET6045_PAR_CNT4_CLR_ENA	568h	CLR入力時CNT4クリア	0/1	0:無効, 1:有効

識別子	値	内容	設定範囲	設定値／取得値 (網掛け:デフォルト(リセット時))
MNET6045_PAR_CMP1_CNT	570h	CMP1 比較カウンタ	0～3	0:カウンタ1 1:カウンタ2 2:カウンタ3 3:カウンタ4
MNET6045_PAR_CMP1_MODE	571h	CMP1 比較方法	0～7	0:コンパレータ機能OFF 1:CMP1=比較ｶｳﾝﾀ (ｶｳﾝﾄ方向無関係) 2:CMP1=比較カウンタ (ｶｳﾝﾄｱｯﾌﾟ中) 3:CMP1=比較カウンタ (ｶｳﾝﾄﾀﾞｳﾝ中) 4:CMP1>比較カウンタ 5:CMP1<比較カウンタ 6:+ソフトELとして利用(CMP1<ｶｳﾝﾀ1) 7:コンパレータ機能OFF
MNET6045_PAR_CMP1_PROC	572h	CMP1 成立処理	0～3	0:処理なし 1:即停止 2:減速停止
MNET6045_PAR_CMP1_DATA	573h	コンパレータ 1		コンパレータ1データ 0
MNET6045_PAR_CMP1_RING	574h	CNT1 リングカウント	0/1	0:通常,1:リングカウント
MNET6045_PAR_CMP2_CNT	578h	CMP2 比較カウンタ	0～3	0:カウンタ1 1:カウンタ2 2:カウンタ3 3:カウンタ4
MNET6045_PAR_CMP2_MODE	579h	CMP2 比較方法	0～7	0:コンパレータ機能OFF 1:CMP2=比較ｶｳﾝﾀ (ｶｳﾝﾄ方向無関係) 2:CMP2=比較カウンタ (ｶｳﾝﾄｱｯﾌﾟ中) 3:CMP2=比較カウンタ (ｶｳﾝﾄﾀﾞｳﾝ中) 4:CMP2>比較カウンタ 5:CMP2<比較カウンタ 6:-ソフトELとして利用(CMP2>ｶｳﾝﾀ1) 7:コンパレータ機能OFF
MNET6045_PAR_CMP2_PROC	57Ah	CMP2 成立処理	0～3	0:処理なし 1:即停止 2:減速停止
MNET6045_PAR_CMP2_DATA	57Bh	コンパレータ 2		コンパレータ2データ 0
MNET6045_PAR_CMP2_RING	57Ch	CNT2 リングカウント	0/1	0:通常,1:リングカウント
MNET6045_PAR_CMP3_CNT	580h	CMP3 比較カウンタ	0～3	0:カウンタ1 1:カウンタ2 2:カウンタ3 3:カウンタ4
MNET6045_PAR_CMP3_MODE	581h	CMP3 比較方法	0～7	0:コンパレータ機能OFF 1:CMP3=比較ｶｳﾝﾀ (ｶｳﾝﾄ方向無関係) 2:CMP3=比較カウンタ (ｶｳﾝﾄｱｯﾌﾟ中) 3:CMP3=比較カウンタ (ｶｳﾝﾄﾀﾞｳﾝ中) 4:CMP3>比較カウンタ 5:CMP3<比較カウンタ 6:設定禁止 7:コンパレータ機能OFF
MNET6045_PAR_CMP3_PROC	582h	CMP3 成立処理	0～3	0:処理なし 1:即停止 2:減速停止
MNET6045_PAR_CMP3_DATA	583h	コンパレータ 3		コンパレータ3データ 0

Motionnet®
Remote I/O & Remote Motion

識別子	値	内容	設定範囲	設定値／取得値 (網掛け：デフォルト(リセット時))
MNET6045_PAR_CMP4_CNT	588h	CMP4 比較カウンタ	0～3	0:カウンタ1 1:カウンタ2 2:カウンタ3 3:カウンタ4
MNET6045_PAR_CMP4_MODE	589h	CMP4 比較方法	0～15	0:コンパレータ機能OFF 1:CMP4=比較カウンタ (ｶｳﾝﾄ方向無関係) 2:CMP4=比較カウンタ (ｶｳﾝﾄｱｯﾌﾟ中) 3:CMP4=比較カウンタ (ｶｳﾝﾄﾀﾞｳﾝ中) 4:CMP4>比較カウンタ 5:CMP4<比較カウンタ 6:-ELとして利用(CMP2 > カウンタ1) 8:IDX(同期)信号出力 (ｶｳﾝﾄ方向無関係) 9:IDX(同期)信号出力 (ｶｳﾝﾄｱｯﾌﾟ中) 10:IDX(同期)信号出力 (ｶｳﾝﾄﾀﾞｳﾝ中) 他:コンパレータ機能OFF
MNET6045_PAR_CMP4_PROC	58Ah	CMP4 成立処理	0～3	0:処理なし 1:即停止 2:減速停止
MNET6045_PAR_CMP4_DATA	58Bh	コンパレータ 4		コンパレータ4データ 0
MNET6045_PAR_CMP4_IDX	58Ch	同期信号出力	0/1	0:CNT4=CMP2の間IDX信号を出力 1:CNT4がｶｳﾝﾄにより0になった時にIDX信号を出力
MNET6045_PAR_CMP5_CNT	590h	CMP5 比較カウンタ	0～5	0:カウンタ1 1:カウンタ2 2:カウンタ3 3:カウンタ4 4:位置決めカウンタ 5:現在速度
MNET6045_PAR_CMP5_MODE	591h	CMP5 比較方法	0～7	0:コンパレータ機能OFF 1:CMP5=比較カウンタ (ｶｳﾝﾄ方向無関係) 2:CMP5=比較カウンタ (ｶｳﾝﾄｱｯﾌﾟ中) 3:CMP5=比較カウンタ (ｶｳﾝﾄﾀﾞｳﾝ中) 4:CMP5>比較カウンタ 5:CMP5<比較カウンタ 他:コンパレータ機能OFF
MNET6045_PAR_CMP5_PROC	592h	CMP5 成立処理	0～3	0:処理なし 1:即停止 2:減速停止
MNET6045_PAR_CMP5_DATA	593h	コンパレータ 5		コンパレータ5データ 0

Motionnet®
Remote I/O & Remote Motion

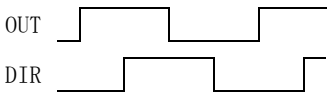
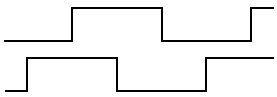
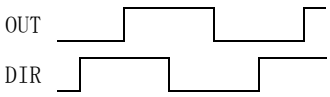
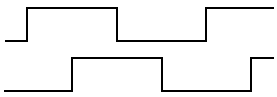
識別子	値	内容	設定範囲	設定値／取得値 (網掛け：デフォルト(リセット時))
MNET6045_PAR_LTC_MODE	598h	ラッチタイミング	0～3	0:LTC入力時 1:ORG入力時 2:CMP4成立時 3:CMP5成立時
MNET6045_PAR_SPD_LTC	599h	CNT3 で速度ラッチ	0/1	0:無効,1:有効
MNET6045_PAR_LTC_INP_DIS	59Ah	ハードラッチ	0/1	0:有効,1:無効
MNET6045_PAR_CNT1_LTC_RST	59Bh	ｶｳﾝﾀ 1 ﾗｯﾁ時ｶｳﾝﾀﾘｾｯﾄ	0/1	0:無効,1:有効
MNET6045_PAR_CNT2_LTC_RST	59Ch	ｶｳﾝﾀ 2 ﾗｯﾁ時ｶｳﾝﾀﾘｾｯﾄ	0/1	0:無効,1:有効
MNET6045_PAR_CNT3_LTC_RST	59Dh	ｶｳﾝﾀ 3 ﾗｯﾁ時ｶｳﾝﾀﾘｾｯﾄ	0/1	0:無効,1:有効
MNET6045_PAR_CNT4_LTC_RST	59Eh	ｶｳﾝﾀ 4 ﾗｯﾁ時ｶｳﾝﾀﾘｾｯﾄ	0/1	0:無効,1:有効
MNET6045_PAR_SET_EVTINT	5A1h	イベント割込設定		8-2-18 参照 0
MNET6045_PAR_SET_CSTA_TYPE	5B0h	CSTA入力仕様	0/1	0:レベルトリガ, 1:エッジトリガ
MNET6045_PAR_SET_CSTP_PROC	5B1h	CSTP入力時処理	0/1	0:即停止, 1:減速停止
MNET6045_PAR_SET_SYNC_OUT	5B2h	内部同期信号出力タイミング	0～15	0:内部同期信号出力OFF 1:コンパレータ1条件成立時 2:コンパレータ2条件成立時 3:コンパレータ3条件成立時 4:コンパレータ4条件成立時 5:コンパレータ5条件成立時 8:加速開始時 9:加速終了時 10:減速開始時 11:減速終了時 その他:内部同期信号出力OFF
MNET6045_PAR_SET_SYNC_INP	5B3h	内部同期信号スタート入力選択	0～3	0:軸0が出力した内部同期信号 1:軸1が出力した内部同期信号 2:軸2が出力した内部同期信号 3:軸3が出力した内部同期信号
MNET6045_PAR_SD_ENA	5C0h	SD 入力	0/1	0:無効, 1:有効(注1)
MNET6045_PAR_INP_ENA	5C1h	INP 入力	0/1	0:無効, 1:有効(注1)
MNET6045_PAR_PCS_ENA	5C2h	PCS 入力	0/1	0:無効, 1:有効(注1)
MNET6045_PAR_INTP_CONST	5C3h	合成速度一定	0/1	0:無効, 1:有効(注1)(注2)
MNET6045_PAR_MSY_MODE	5C4h	同期スタートモード	0/1	0:無効(注1) 1:同時スタート(CSTA)信号待ち 2:内部同期スタート信号待ち 3:指定軸の停止待ち
MNET6045_PAR_MSY_SAX	5C5h	指定軸停止同期スタート設定	0～15	停止確認する軸パターンを4ビットで設定します。デフォルト:0h(注1) ビット0:軸0の停止を確認します ビット1:軸1の停止を確認します ビット2:軸2の停止を確認します ビット3:軸3の停止を確認します
MNET6045_PAR_CSTP_ENA	5C6h	CSTP 入力	0/1	0:無効, 1:有効(注1)
MNET6045_PAR_AUTO_CSTP_ENA	5C7h	CSTP 自動出力	0/1	0:無効, 1:有効(注1)
MNET6045_PAR_FHCR_DIS	5C8h	FH 補正	0/1	0:有効, 1:無効(注1)
MNET6045_PAR_SDP_SOFTCALC	5C9h	スローダウンポイント演算	0/1	0:自動切換, 1:ソフト演算固定(注1)

注 1:これらのパラメータ設定は動作起動時(あるいは次動作書込み時)に適用されます。
(動作中の移動には適用されません)

注 2:全軸共通のデータとなります。(軸番号の指定は無視されます)

出力パルスタイプ

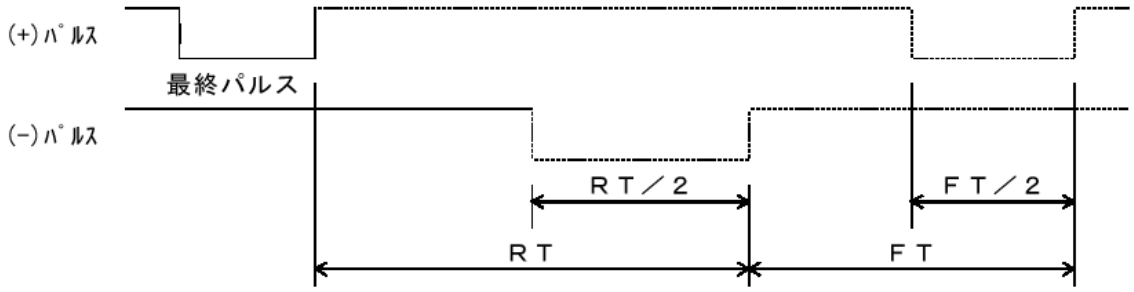
出力パルス仕様を設定します。

設定/取得値	（＋）方向動作時		（－）方向動作時	
	PULS 信号	DIR 信号	PULS 信号	DIR 信号
0		High		Low
1		High		Low
2		Low		High
3		Low		High
4		High	High	
5				
6				
7		Low	Low	

振動抑制

項目	設定内容
振動抑制１	下図のＲＴの時間を設定します。設定単位は本ボード基準クロック周期(19.6608MHZ)の32倍(約1.6μｓです)
振動抑制２	下図のＦＴの時間を設定します。設定単位は本ボード基準クロック周期(19.6608MHZ)の32倍(約1.6μｓです)

下図の最終パルス以降が振動抑制機能により付加されるパルスです



Motionnet®
Remote I/O & Remote Motion

8−3−3−11. DLL 内部パラメータ項目番号定義

MNET6045SetDLLPar 関数使用時に対象のパラメータ項目を指定します。

識別子	値	内容	設定範囲	設定値／取得値（網掛け：デフォルト）
MNET6045_DLLPAR_PRECHK_DIS	5E1h	軸動作前確認省略	0/1	0:省略しない,1:省略
MNET6045_DLLPAR_ADCCR_DIS	5E2h	加減速設定自動補正	0/1	0:補正する,1:補正しない
MNET6045_DLLPAR_ENDLESS_ARC	5E3h	無限円弧指令時処理	0/1	0:無限動作,1:エラーとする(注1)

注 1:ボード単位の項目となります。指定時,軸番号には 0～3 の任意の値を設定ください。

8−3−3−12. 戻り値

(1)軸番号が付加されない戻り値

識別子	値	内容
MNET6045_ERR_AXNO	511Fh	軸番号指定不正 軸番号の指定が正しくない(0以下または3以上である)場合に返されます。
MNET6045_ERR_AXPTN	513Fh	軸パターン指定不正 軸パターンの指定が正しくない(0以下またはFh以上である)場合、円弧関数指令時に2つ以上の軸を含む軸パターン指定した場合などに返されます。
MNET6045_ERR_SET_STAMODE	532Fh	スタートモード指定不正 スタートモードの値が選択可能な値でありません。
MNET6045_ERR_SET_PREREG	533Fh	プリレジ指定不正 プリレジ指定の値が選択可能な値でありません。
MNET6045_ERR_SET_INTERPOLATION	534Fh	補間指令データ不正 {直線補間指令において合成移動量が0である場合に返されます。}
MNET6045_ERR_SET_INTP_ADC_DATA	535Fh	補間時加減速データ設定不正 {合成速度一定制御ONの場合に加速データと減速データが異なる場合に返されます。}
MNET6045_ERR_SET_ARC_POS_DATA	536Fh	円弧補間時位置データ設定エラー 円の半径が1パルス以下、あるいは無限円弧指令をエラーとして扱う設定の場合に返されます。
MNET6045_ERR_PCS_MODE	537Fh	PCSモードでは補間不可 PCS有効の状態で補間動作を指令した場合に返されます。
MNET6045_ERR_SET_SYNC_INTP	538Fh	補間時同期スタート設定不一致 補間指令時に補間に含まれる軸の同期スタート設定が一致しない場合に返されます。
MNET6045_ERR_FUNC_UNAVAILABLE	539Fh	関数利用制限(動作状態など) 以下の様な利用できない条件で指令された場合に返されます。 ・ スタートモードがFH定速または加減速でパルサー関数を指令された場合 ・ ELを使った原点復帰モードで原点サーチ関数が指令された場合 ・ 不適切な速度設定で原点サーチ関数が指令された場合 ・ プリレジスタに次動作が設定されている状態で位置オーバーライド関数が指令された場合 ・ Ptp, PtPInc動作以外の時に位置オーバーライド関数が指令された場合
MNET6045_ERR_FAILED_OVRSPD	53AFh	速度オーバライド失敗 減速中の場合、残パルスが足りない場合、関数処理中にプリレジスタがシフトした場合など速度オーバーライドが失敗した際に返されます。
MNET6045_ERR_GETDATA_FAILED	53BFh	データ取得失敗 関数処理中にプリレジスタがシフトした場合に返されます。
MNET6045_ERR_ACC_DATA_MIXED	53CFh	加速無し速度設定の軸と加速有り速度設定の軸に対して、まとめて移動指令を与えた場合に返されます。
MNET6045_ERR_FAILED_OVRPOS	53DFh	位置オーバーライド失敗 残パルスが足りないなど、位置オーバーライドに失敗した場合に返されます。
MNET_ERR_6045_FAILED_OPENFILE	53EFh	パラメータファイルのオープン失敗
MNET_ERR_6045_INVALID_FILEDATA	540Fh	パラメータファイルデータ不正 以下のような場合に返されます。 ・ 項目数が正しくない場合 ・ 文字からデータへの変換不良 ・ 基板型式とパラメータファイルが適合していない
MNET_ERR_DEV_NOT_6045	53FFh	デバイスが４軸(6045)モーションボードでない

(2)軸番号が付加される戻り値

識別子	値	内容
MNET6045_ERR_SET_SPD_DATA	5800h	速度指定エラー 指定された速度が0-6553500の範囲を超える場合、また速度倍率を固定設定している場合に、速度がレンジの範囲外である場合に返されます。
MNET6045_ERR_CALC_SPD_DATA	5810h	速度レート計算結果エラー {速度計算に失敗した場合に返されます。}
MNET6045_ERR_SET_ADC_DATA	5820h	加減速度指定エラー 加減速の指定データが正しくない(0以下)場合に返されます。
MNET6045_ERR_CALC_ACC_DATA	5830h	加速レート計算結果エラー 指定された加速データに基づいて計算した加速データ値がLSIの設定範囲を超える場合に返されます。 (通常は自動的に最大値、最小値が適用されますが、自動補正OFF設定である場合はエラーとなります)
MNET6045_ERR_CALC_DEC_DATA	5840h	減速レート計算結果エラー 指定された減速データに基づいて計算した減速データ値がLSIの設定範囲を超える場合に返されます。 (通常は自動的に最大値、最小値が適用されますが、自動補正OFF設定である場合はエラーとなります)
MNET6045_ERR_SET_SSECT	5850h	S字範囲指定エラー S字範囲指定が0以下の場合に返されます。
MNET6045_ERR_SET_ADCSPC	5860h	加減速方法指定エラー 加減速指定方法の値が選択可能な値でありません。
MNET6045_ERR_SET_SMODE	5870h	S字モード指定エラー S字モード指定の値が選択可能な値でありません。
MNET6045_ERR_SET_DIR	5880h	方向指定エラー 方向指定の値が選択可能な値でありません。
MNET6045_ERR_STATUS_AXIS_MOVING	5890h	動作中 当該軸が動作中である場合には受け付けられない関数が実行された場合に返されます。
MNET6045_ERR_STATUS_PREREG_FULL	58A0h	プリレジスタ満杯 プリレジスタ満杯のため次動作書き込みは受け付けられません
MNET6045_ERR_STATUS_ALM	58B0h	動作前確認(ALM) 軸動作関数の動作前確認でALM入力がONしているため軸動作関数は実行されません。(動作前確認はMNET6045SetDLLPar関数で省略可)
MNET6045_ERR_STATUS_EMG	58C0h	動作前確認(EMG) 軸動作関数の動作前確認でEMG入力がONしているため軸動作関数は実行されません。(動作前確認はMNET6045SetDLLPar関数で省略可)
MNET6045_ERR_STATUS_EL	58D0h	動作前確認(EL) 軸動作関数の動作前確認でEL入力がONしているため軸動作関数は実行されません。(動作前確認はMNET6045SetDLLPar関数で省略可)
MNET6045_ERR_STATUS_ALREADY	58E0h	動作前確認(すでに目的の状態にある) 軸動作関数の動作前確認で信号入力が既に目的の状態にあるため軸動作関数は実行されません。 (動作前確認はMNET6045SetDLLPar関数で省略可)
MNET6045_ERR_STATUS_CSTP	58F0h	動作前確認(CSTP) 軸動作関数の動作前確認でCSTP入力がONしているため軸動作関数は実行されません。(動作前確認はMNET6045SetDLLPar関数で省略可)

識別子	値	内容
MNET6045_ERR_STATUS_SOFTLIMIT	5910h	動作前確認(ソフトリミット) 軸動作関数の動作前確認で指令値がソフトリミット値を越えているため軸動作関数は実行されません。 (動作前確認はMNET6045SetDLLPar関数で省略可)
MNET6045_ERR_STATUS_SERR	5920h	動作前確認(エラー割込あり) 軸動作関数の動作前確認でエラー割込が確認されたため軸動作関数は実行されません。 次動作書き込みの場合、エラー割込を示すSERRビットがONしているときは指令は受け付けられません。 MNET6045GetINTFactor関数などでエラー割込要因データを読み出すことでSERRはOFFします。
MNET6045_ERR_STATUS_PREREG_MODE	5930h	プリレジ継続不可 原点復帰動作やMNET6045Continue関数など位置決めではない動作の後、あるいは移動指令がエラーになった後には次動作書込みを行なうことができません。
MNET6045_ERR_SPDDATATA_INVALID	5940h	速度データ無効 指定軸の速度データが設定されていません。 またMNET6045SetSpeed関数で設定した速度データは、補間指令を受け付けることで無効となります。非補間指令起動にあたっては再設定が必要となります。

戻り値への軸番号の付加について

値の 16 進値最下位桁が F 以外の場合、値に軸番号(0～3)が最下位桁に付加(OR)されます。

[例 1]
値　5892h は 5890h(エラー内容を示すコード) + 2(軸番号)を表し、軸 2 の動作中エラーを示します。
[例 2]
値　58A0h は 58A0h(エラー内容を示すコード) + 0(軸番号)を表し、軸 0 のプリレジ満杯を示します。

9. 関数インデックス

9−1. センターボード機種別関数

9−1−1. PPCI-L112 ボード関数

No.	関数名称	機能
1	PPCI_L112Open	ボードオープン
2	PPCI_L112Close	ボードクローズ
3	PPCI_L112GetParaIn	パラレル入力読出し
4	PPCI_L112GetParaOut	パラレル出力読出し
5	PPCI_L112SetParaOut	パラレル出力書込み
6	PPCI_L112SetCommSpeed	通信速度設定

9−1−2. NPMCMNET-I/0104 ボード関数

No.	関数名称	機能
1	NPMCMNETOpen	ボードオープン
2	NPMCMNETClose	ボードクローズ
3	NPMCMNETGetParaIn	パラレル入力読出し
4	NPMCMNETGetParaOut	パラレル出力読出し
5	NPMCMNETSetParaOut	パラレル出力書込み
6	NPMCMNETSetCommSpeed	通信速度設定

9−1−3. MNET-PUSB3601 ボード関数

No.	関数名称	機能
1	MNETPUsbOpen	ボードオープン
2	MNETPUsbClose	ボードクローズ
3	MNETPUsbSetCommSpeed	通信速度設定

9−2. センターボード共通関数

No.	関数名称	機能
1	MNETSysComm	システム通信
2	MNETIOCommStart	I/O通信スタート
3	MNETIOCommStop	I/O通信ストップ
4	MNETCenterReset	センターデバイスリセット
5	MNETGetBRKF	ブレークフレーム有無確認
6	MNETGetEIOE	I/O通信エラー確認
7	MNETGetIOPC	入力ポート変化割込確認
8	MNETGetSBSY	I/O通信中確認
9	MNETGetDevType	ローカルデバイス種類読出し

9−3. ローカルボード共通関数

No.	関数名称	機能
1	MNETSetLocalData	ローカルデバイスデータ書込み
2	MNETGetLocalData	ローカルデバイスデータ読出し
3	MNETSetDLLPar	DLL内部パラメータ設定

9−4. ローカルボード機種別関数

9−4−1. ローカル 1 軸モーションボード関数

No.	関数名称	機能
1	MNETM1SetSpeed	速度設定
2	MNETM1Inc	Inc 移動
3	MNETM1Abs	Abs 移動
4	MNETM1Continue	連続移動
5	MNETM1Org	原点復帰移動
6	MNETM1EscOrg	原点脱出移動
7	MNETM1ToEL	EL まで移動
8	MNETM1EscEL	EL 脱出移動
9	MNETM1Ez	EZ カウント移動
10	MNETM1Stop	軸停止
11	MNETM1GetSpeed	軸速度取得
12	MNETM1GetCounter	カウンタ値取得
13	MNETM1SetCounter	カウンタ値設定
14	MNETM1GetLtc	ラッチデータ取得
15	MNETM1GetBitDataStruct	ビットデータ一括取得
16	MNETM1GetBitData	ビットデータ取得
17	MNETM1SetBitData	ビットデータ設定
18	MNETM1FixSpeedRange	速度倍率固定
19	MNETM1UnFixSpeedRange	速度倍率固定解除
20	MNETM1OutCsta	CSTA 信号出力
21	MNETM1OutCstp	CSTP 信号出力
22	MNETM1OvrideSpeed	速度変更
23	MNETM1OvridePos	目標位置オーバーライド
24	MNETM1ResetSEND	SEND(位置決め完了)ビットリセット
25	MNETM1SetPar	パラメータ設定
26	MNETM1GetPar	パラメータ取得
27	MNETM1SearchORG	原点サーチ移動
28	MNETM1GetREST	エラー割り込み要因取得
29	MNETM1GetRIST	イベント割り込み要因取得
30	MNETM1Restart	位置決めリスタート
31	MNETM1SoftPCS	PCS 入力代行コマンド発行
32	MNETM1SoftLTC	LTC 入力代行コマンド発行
33	MNETM1Reset	モーションデバイスリセット
34	MNETM1SetDLLPar	DLL 内部パラメータ設定
35	MNETM1ReadParFile	パラメータファイル読出し、デバイスに設定
36	MNETM1WriteParFile	デバイスより取得、パラメータファイル書込み

9−4−2. ローカル４軸(6143)モーションボード関数

No.	関数名称	機能
1	MNETM4SetSpeed	速度設定 (軸パターン指定)
2	MNETM4PtpInc	PtpInc移動 (軸パターン指定)
3	MNETM4LineInc	直線補間Inc移動 (軸パターン指定)
4	MNETM4Ptp	Ptp移動 (軸パターン指定)
5	MNETM4Line	直線補間移動 (軸パターン指定)
6	MNETM4Continue	連続移動 (軸パターン指定)
7	MNETM4Org	原点復帰移動 (軸パターン指定)
8	MNETM4SetSpeedNo	速度設定 (軸番号指定)
9	MNETM4PtpIncNo	PtpInc移動 (軸番号指定)
10	MNETM4PtpNo	Ptp移動 (軸番号指定)
11	MNETM4ContinueNo	連続移動 (軸番号指定)
12	MNETM4OrgNo	原点復帰移動 (軸番号指定)
13	MNETM4StopChk	軸停止確認
14	MNETM4GetREST	エラー割り込み要因取得
15	MNETM4GetRIST	イベント割り込み要因取得
16	MNETM4CancelPreReg	次動作キャンセル
17	MNETM4Stop	軸停止
18	MNETM4SetBitData	ビットデータ設定
19	MNETM4GetBitData	ビットデータ取得
20	MNETM4GetBitDataStruct	ビットデータ一括取得
21	MNETM4GetSpeed	軸速度取得
22	MNETM4GetCounter	カウンタ値取得
23	MNETM4SetCounter	カウンタ値設定
24	MNETM4GetLtc	ラッチデータ取得
25	MNETM4SoftLTC	LTC入力代行コマンド発行
26	MNETM4SoftPCS	PCS入力代行コマンド発行
27	MNETM4OutCsta	CSTA信号出力
28	MNETM4OutCstp	CSTP信号出力
29	MNETM4SetPar	パラメータ設定
30	MNETM4GetPar	パラメータ取得
31	MNETM4SetDLLPar	DLL内部パラメータ設定
32	MNETM4Reset	パルスコントロールLSIリセット
33	MNETM4ReadParFile	パラメータファイル読出し、デバイスに設定
34	MNETM4WriteParFile	デバイスより取得、パラメータファイル書込み

9−4−3. ローカル４軸(6045)モーションボード関数

No.	関数名称	機能
1	MNET6045SetSpeed	(非補間動作)速度設定 (軸パターン指定)
2	MNET6045PtpInc	Ptp移動(Inc指定) (軸パターン指定)
3	MNET6045Ptp	Ptp移動(Abs指定) (軸パターン指定)
4	MNET6045LineInc	直線補間移動(Inc指定) (軸パターン指定)
5	MNET6045Line	直線補間移動(Abs指定) (軸パターン指定)
6	MNET6045ArcInc	円弧補間移動(Inc指定) (軸パターン指定)
7	MNET6045Arc	円弧補間移動(Abs指定) (軸パターン指定)
8	MNET6045Continue	連続移動 (軸パターン指定)
9	MNET6045Org	原点復帰移動 (軸パターン指定)
10	MNET6045EscOrg	原点脱出移動 (軸パターン指定)
11	MNET6045SearchOrg	原点サーチ移動 (軸パターン指定)
12	MNET6045ToEL	ELまで移動 (軸パターン指定)
13	MNET6045EscEL	EL脱出移動 (軸パターン指定)
14	MNET6045Ez	EZカウント移動 (軸パターン指定)
15	MNET6045Stop	軸停止 (軸パターン指定)
16	MNET6045SetSpeedNo	(非補間動作)速度設定 (軸番号指定)
17	MNET6045PtpIncNo	Ptp移動(Inc指定) (軸番号指定)
18	MNET6045PtpNo	Ptp移動(Abs指定) (軸番号指定)
19	MNET6045ContinueNo	連続移動 (軸番号指定)
20	MNET6045OrgNo	原点復帰移動 (軸番号指定)
21	MNET6045EscOrgNo	原点脱出移動 (軸番号指定)
22	MNET6045SearchOrgNo	原点サーチ移動 (軸番号指定)
23	MNET6045ToELNo	ELまで移動 (軸番号指定)
24	MNET6045EscELNo	EL脱出移動 (軸番号指定)
25	MNET6045EzNo	EZカウント移動 (軸番号指定)
26	MNET6045StopNo	軸停止 (軸番号指定)
27	MNET6045GetBitStruct	ビットデータ一括取得
28	MNET6045GetBit	ビットデータ取得
29	MNET6045SetBit	ビットデータ設定
30	MNET6045GetSpeed	現在速度取得
31	MNET6045GetCounter	カウンタ値取得
32	MNET6045SetCounter	カウンタ値設定
33	MNET6045Reset	LSIリセット
34	MNET6045CancelPreReg	次動作キャンセル
35	MNET6045StopChk	軸停止確認
36	MNET6045SetDLLPar	DLLパラメータ設定
37	MNET6045SetPar	パラメータ設定
38	MNET6045GetPar	パラメータ取得
39	MNET6045FixSpeedRange	速度レンジ固定設定
40	MNET6045UnFixSpeedRange	速度レンジ固定設定解除
41	MNET6045SetFASpeed	補正速度設定
42	MNET6045OvrideSpd	速度オーバーライド
43	MNET6045GetRemain	移動残パルス取得
44	MNET6045GetLtc	ラッチデータ取得
45	MNET6045SoftLtc	ラッチ入力代行コマンド発行
46	MNET6045SoftPcs	PCS入力代行コマンド発行
47	MNET6045OutCsta	同時スタートコマンド発行
48	MNET6045OutCstp	同時ストップコマンド発行

Motionnet® Remote I/O & Remote Motion

No.	関数名称	機能
49	MNET6045GetTermAbs	最終到達位置データ取得
50	MNET6045OvridePos	目標位置オーバーライド
51	MNET6045ReadParFile	パラメータファイル読出し
52	MNET6045WriteParFile	パラメータファイル書込み
53	MNET6045PulserPtpInc	パルサー同期Ptp移動(Inc指定) (軸パターン指定)
54	MNET6045PulserPtp	パルサー同期Ptp移動(Abs指定) (軸パターン指定)
55	MNET6045PulserLineInc	パルサー同期直線補間移動(Inc指定) (軸パターン指定)
56	MNET6045PulserLine	パルサー同期直線補間移動(Abs指定) (軸パターン指定)
57	MNET6045PulserArcInc	パルサー同期円弧補間移動(Inc指定) (軸パターン指定)
58	MNET6045PulserArc	パルサー同期円弧補間移動(Abs指定) (軸パターン指定)
59	MNET6045PulserContinue	パルサー同期連続移動 (軸パターン指定)
60	MNET6045PulserPtpIncNo	パルサー同期Ptp移動(Inc指定) (軸番号指定)
61	MNET6045PulserPtpNo	パルサー同期Ptp移動(Abs指定) (軸番号指定)
62	MNET6045PulserContinueNo	パルサー同期連続移動 (軸番号指定)
63	MNET6045GetINTFactor	割込要因取得

10. 基本関数インデックス

10−1. センターデバイス基本関数

No.	関数名称	機能
1	MNETSetCenterCmd	センターデバイスコマンド書込み
2	MNETGetCenterSts	センターデバイスステータス読出し
3	MNETGetCenterIntSts	センターデバイス割込ステータス読出し
4	MNETWriteCenterRegister	センターデバイスレジスタ書込み
5	MNETReadCenterRegister	センターデバイスレジスタ読出し

10−2. 1軸モーションデバイス基本関数

No.	関数名称	機能
1	MNETM1WriteRegister	モーションデバイスレジスタ書込み
2	MNETM1ReadRegister	モーションデバイスレジスタ読出し
3	MNETM1WriteCommand	モーションデバイスコマンド書込み

10−3. 4軸(6045)モーションデバイス基本関数

No.	関数名称	機能
1	MNET6045WReg	モーションデバイスレジスタ書込み
2	MNET6045RReg	モーションデバイスレジスタ読出し
3	MNET6045WCmd	モーションデバイスコマンド書込み
4	MNET6045WPort	モーションデバイスポート書込み
5	MNET6045RSts	モーションデバイスステータス読出し

注意	性能、品質の向上等にともない、お断り無しに掲載事項を変更させていただくことがありますので、あらかじめご了承ください。

NPM 日本パルスモーター株式会社

本　　社　〒113-0033　東京都文京区本郷 2-16-13　電話　03-3813-8841(代)　FAX. 03-3813-8550
大阪営業所　〒552-0007　大阪市港区弁天 1-2-1-2314　電話　06-6576-8330　FAX. 06-6576-8335
ＵＲＬ：　http://www.pulsemotor.com/

２０１５年２月発行